

AD-A168 197

DEVELOPMENT OF ALGORITHMS AND HARDWARE FOR  
STRUCTURED-LIGHT VISION(U) GEORGE WASHINGTON UNIV  
WASHINGTON DC DEPT OF ELECTRICAL ENGIN.

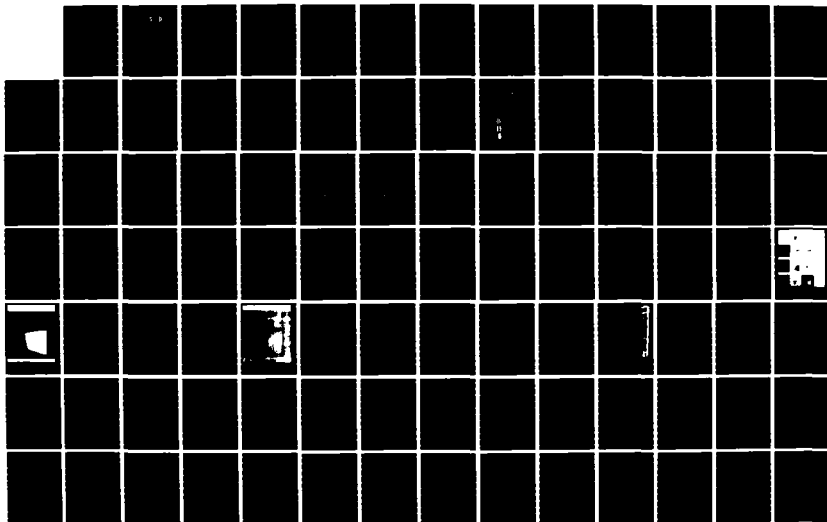
1/4

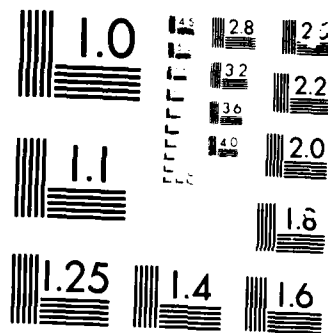
UNCLASSIFIED

M H LOEW ET AL. 1986 N00014-83-K-0578

F/G 28/6

NL

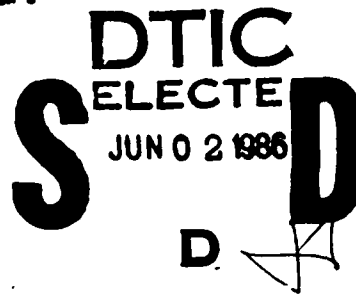




MICROGRAPH

1010A

AD-A168 197



DEVELOPMENT OF ALGORITHMS AND HARDWARE  
FOR STRUCTURED-LIGHT VISION

Final Report, Contract No. N00014-83-K-0578  
for the  
Office of Naval Research

Department of Electrical Engineering and Computer Science  
George Washington University  
Washington, DC 20052

**DISTRIBUTION STATEMENT A**  
Approved for public release;  
Distribution Unlimited

86 5 5 012

# TABLE OF CONTENTS

|  |       |
|--|-------|
| List of Figures (Part II.B).....                                       | i     |
| List of Figures (Part III.B).....                                      | iii   |
| List of Plates.....  | iv    |
| List of Tables.....  | v     |
| <br>I. Introduction and Goals.....                                     | <br>1 |
| II. Background, approach, and role of Computer Vision.....             | 2     |
| A. Role and Competence of Structured-light systems.....                | 2     |
| B. Specific Approach.....  | 7     |
| 1. The Basic Algorithm.....  | 7     |
| 2. 'Visible' Field of View.....  | 20    |
| 3. Camera-Projector Set up.....  | 24    |
| 4. Light source and slit.....  | 24    |
| 5. Hardware-Algorithm Modification.....                                | 35    |
| 6. Look-Up Tables' Restriction of Resolution and<br>Dynamic Range..... | 37    |
| 7. Dark Objects' Range.....  | 42    |
| 8. Mask Adjustment.....  | 42    |
| III. Topics Investigated.....  | 43    |
| A. Hardware.....   | 43    |
| B. Software.....   | 44    |
| 1. Introduction.....   | 44    |
| 2. Simulation System.....  | 45    |
| 3. Range Data Recognition.....   | 51    |
| 4. Object matching.....  | 59    |
| 5. Results and Conclusions.....  | 67    |
| IV. Conclusions and Implications for the Navy.....                     | 67    |
| Appendix A.....  | 70    |
| Appendix B.....  | 81    |
| Appendix C.....  | 134   |
| Appendix D.....  | 177   |
| Appendix E.....  | 185   |
| Appendix F.....  | 223   |

|                                     |
|-------------------------------------|
| <input checked="" type="checkbox"/> |
| <input type="checkbox"/>            |
| <input type="checkbox"/>            |

*Det. on file*



| Availability Codes |                      |
|--------------------|----------------------|
| Dist               | Avail and/or Special |
| A-1                |                      |



## LIST OF FIGURES

## PART II.B -- SOFTWARE

|            |  |    |
|------------|--|----|
| Figure 1   | An Object and its distances from a reference plane.....  | 8  |
| Figure 2   | Triangulation geometry: R may be found if $\alpha$ , $\beta$ and D are known.....                                  | 9  |
| Figure 3   | Mask with m bars each of size a.....   | 10 |
| Figure 4   | Using the mask to measure $\alpha$ .....   | 11 |
| Figure 5   | A ranging system, with source P, mask M, and CID camera focused at C.....  | 13 |
| Figure 6   | Masks of alternating transparent and opaque sections, and the reference mask.....                                  | 14 |
| Figure 7a  | Construction of the bit plane: the reference image, to account for incident intensity and object reflectivity..... | 16 |
| Figure 7b  | Bit plane level one; result of two-bar mask.....   | 16 |
| Figure 7c  | Bit plane level two; result of four-bar mask.....  | 17 |
| Figure 7d  | Collecting masks' results into a single code.....  | 17 |
| Figure 8   | Using the camera to measure $\beta$ .....  | 19 |
| Figure 9   | Defining the visible field and geometry of the definition of resolution.....                                       | 21 |
| Figure 10  | Correct orientation of apparatus for measuring range.....  | 25 |
| Figure 11  | An incorrect orientation; $\alpha_2$ will be measured, which does not contribute to range information.....         | 26 |
| Figure 12a |  |    |

|   |    |
|---|----|
| A wide slit yields ambiguous regions.....                                 | 27 |
| Figure 12b  |    |
| A perfect slit yields only dichotomy.....                                 | 28 |
| Figure 13   |    |
| Adjustable slit used in the ranging system.....                           | 30 |
| Figure 14   |    |
| Geometry for effect of gray-level uncertainty on range<br>resolution..... | 32 |
| Figure 15   |    |
| Geometry for calculation of total false-black area.....                   | 34 |
| Figure 16   |    |
| Construction of correct codes using inversion.....                        | 36 |
| Figure 17   |    |
| Geometry for evaluation of intrinsic range uncertainty....                | 40 |
| Figure 18   |    |
| Another view of range uncertainty.....                                    | 41 |

## LIST OF FIGURES

## III.B - SOFTWARE

|  |    |
|--|----|
| Figure 1   |    |
| Summary of the simulation procedure.....   | 46 |
| Figure 2   |    |
| Flowchart of plane-description procedure.....                                    | 50 |
| Figure 3   |    |
| Slope histograms for identification of distinct planes....                       | 54 |
| Figure 4   |    |
| Effect of limited dynamic range  |    |
| a) cross-section of two planes meeting at shallow angle                          |    |
| b) The corresponding histogram, and its poorly-resolved peaks.....               | 56 |
| Figure 5   |    |
| Enhancement of the example of Fig. 4:  |    |
| a) multiplication by a constant  |    |
| b) smoothing,  |    |
| c) improved histogram.....   | 56 |
| Figure 6   |    |
| Neighborhoods of a peak in the 2-D slope histogram define maxima and minima..... | 57 |
| Figure 7   |    |
| Flowchart of the object-matching procedure.....                                  | 63 |
| Figure 8   |    |
| Example of the object-matching procedure.....                                    | 65 |

## LIST OF PLATES

|   |    |
|---|----|
| Plate 1   |    |
| Results of the operations from Fig. 1.....  | 47 |
| Plate 2   |    |
| A simulated range image, mapped as gray levels. Darker<br>pixels represent smaller distances.....   | 48 |
| Plate 3   |    |
| Images of slope mapped as intensity   |    |
| (a) $dz/dx$ .....   | 52 |
| (b) $dz/dy$ .....   | 53 |
| Plate 4   |    |
| Identification in the range image of the distinct planes;<br>pixels with slopes within the ranges defined<br>by Fig. 6 are given a common gray level..... | 58 |
| Plate 5   |    |
| The three detected planes:  |    |
| (a).....  | 60 |
| (b).....  | 61 |
| (c).....  | 62 |
| They match the original planes  |    |

## LIST OF TABLES

|   |    |
|---|----|
| I. Results of plane-equation determination.....   | 68 |
| II. Results of node-coordinate determination..... | 68 |

DEVELOPMENT OF ALGORITHMS AND HARDWARE  
FOR STRUCTURED-LIGHT VISION

Final Report, Contract No. N00014-83-K-0578  
for the  
Office of Naval Research

by

Murray H. Loew  
Reza Momenan  
Gao Shangkai  
James Hargrove

Department of Electrical Engineering and Computer Science  
George Washington University  
Washington, DC 20052

I. Introduction and Goals

Computer vision is finding increased application in robotics and for shipbuilding, construction, machining, sorting, and inspection. An essential part of a computer vision system is the ability to measure range (distance) to points on the surfaces of three-dimensional objects. This information allows the description of such intrinsic characteristics of objects as orientation, surface condition, and dimensions.

A goal of the work described here was the development, testing, and evaluation of a real-time system for measuring range, suitable for the kinds of applications described above. A second goal was the development of algorithms for description of objects, based on their range-image representations.

In collaboration with the National Bureau of Standards (NBS), George Washington University (GWU) has shown the feasibility, and measured the performance, of a prototype ranging system that uses a technique called structured light; GWU has also derived and implemented algorithms for rapid description of a limited class of objects using only the data that would be supplied by such a system.

## II. Background, approach, and role of computer vision

### A. Role and competence of structured-light systems

Machine vision attempts to extract information about the objects in a scene from the manner in which the objects modulate and reflect the illumination falling on them. In most cases, normal ambient light sources provide minimal constraints which can be employed to simplify this task. Structured light techniques for machine vision employ illumination sources which provide more direct information than is available from normal ambient light. Nonetheless, many approaches to machine vision have attempted to solve the general vision and image understanding problem without recourse to artificially structured illumination. While this presents many interesting problems from an academic standpoint, and is required in those special applications where ambient illumination must be employed, the speedy development of practi-

cal applications in automated assembly and manufacturing could profit from far greater exploitation of structured-light approaches.

In the majority of near-term applications, such as ship-building, construction, machining, welding, assembly, inspection, maintenance, painting, sorting, and materials handling, there is no obstacle to the use of artificially structured illumination. This approach facilitates precision optical ranging and measurement, and greatly simplifies many of the traditional problems of image processing and understanding. Moreover, projectors for structured illumination may be simple and compact, and may be mounted on the manipulator along with the camera.

One of the greatest advantages of the structured light approach is the ability to simplify the determination of depth in the image. This arises from the possibility of knowing the exact camera-relative angle of origin of the ray illuminating any part of the scene. Determination of depth then reduces to a straightforward process of triangulation. In simple applications, a single plane of light may be projected from a manipulator, and its intersection with objects in the field observed by an offset camera. Experiments at NBS have shown this arrangement to be effective for the real-time control of reflexive seek-and-grasp operations, and even for simple object-discrimination tasks.



Additional planes of projected light in such a system enable the direct sensing of three-dimensional tilt of surfaces. When combined with two-dimensional outline images obtained from point-source illumination, such a system is adequate for industrial parts manipulation. With minor modifications (projection onto a surface and examination with a telescope), the system will also work well for automated inspection.

The work described here has made it possible to read the Z-axis coordinate (depth) of any pixel in the scene directly, and thus to define any point of an imaged object immediately in terms of its coordinates in three-dimensional visual space. This ability is central to control of a manipulator that must interact with random objects in real space. However, its most important application is in image understanding, where three-dimensional information is invaluable in resolving ambiguities of projected shape, occlusion, relative size, orientation, segmentation, and the like. So fundamental is this information, that current non-structured (ambient) light techniques devote the majority of their processing time to obtaining three-dimensional descriptions from ambient light cues, or to classifying objects on the basis of two-dimensional cues generated by surface interactions in three dimensions. As a result of the processing bottleneck represented by the difficulty of obtaining three-dimensional information, there are currently no ambient-light vision tech-

Lack of adequate interpreted sensory information is perhaps the greatest single obstacle to the commercial exploitation of robotics. While a full solution to the ambient-light vision problem appears to be many years off, structured-light approaches offer a realizable near-term solution that is applicable in a large variety of important military and industrial settings. It seems likely that any program which could promote the development of commercially available structured-light vision systems would have very favorable consequences for the field of automated manufacturing and in turn for its military and industrial applications. In the next section, we outline such a program.

To be useful in practical applications, a vision system must operate in real time. Essentially, this means that it must be able to accept data as fast as it can be provided, and that it must complete analysis of that data to any given level of complexity as rapidly as analyses at that level are required for uninterrupted operation. Many of the algorithms currently in use for the simpler forms of structured-light vision are within the real-time capacity of standard computer hardware. Others run in "near" real-time, while still others are undergoing development. In all cases, orders-of-magnitude improvement could be expected from application of dedicated specialized hardware, an example of which was built and tested as part of this work.

niques suitably developed for commercial real-time applications except in highly constrained environments (such as flat parts on a conveyor) where three-dimensionality may be ignored safely.

Many techniques have been proposed for obtaining three-dimensional information from ambient light cues. Among them may be mentioned stereopsis, optic flow, shape from shading, and shape from texture. All of these are areas of intense current interest and activity, and many are modeled on psychophysical analogs. However, a practical solution to the very complex problem of three-dimensional ambient light vision may well require a synthesis of many of these approaches, and all of them individually are at best in an experimental state of development. Moreover, all of these approaches are intensely computational in nature, and will require improvements of many orders of magnitude in computing speed to be competitive with structured light techniques that have been demonstrated in real-time vision. They may be adapted in graded levels of complexity according to the task, from simple ranging and parts acquisition to complex image understanding. In many instances these techniques are ready to be moved out of the laboratory and into practical development; in other cases the time to practical development appears significantly shorter than that of any of the ambient light techniques. Many structured light applications are within the real-time computational power of standard computer hardware; others appear to be very close to this goal. This report describes several approaches.

## B. SPECIFIC APPROACH

### B.1 The Basic Algorithm

The basic algorithm of our system employs triangulation. Figure 1 illustrates distances  $R_i$  of points  $x_i$  of object A from a reference plane Q, perpendicular to the plane of the paper and including points C and P. It is possible to reconstruct a 3-D representation of object A with respect to a viewer in plane Q.

The following relationship may be derived from the geometry shown in Figure 2:

$$\frac{1}{R_i} = \frac{1}{D} (\cot \alpha_i + \cot \beta_i)$$

where D is the distance between points C and P, and  $R_i$  is called "range" as defined earlier.

Therefore in order to identify a specific point and hence calculate its range we need to measure or calculate  $\alpha_i$  and  $\beta_i$  and their corresponding cotangents.

Suppose we have a template M such as the one in Figure 3, and place it in front of reference point P at a fixed distance  $f_i$ .

As shown in Figure 4 we can now decide the partition j through which the line (or vector) connecting reference point P to any point  $x_i$  passes, which in turn will give us the value of angle  $\alpha_i$ , hence  $\cot \alpha_i$ :

$$\cot \alpha_i = \frac{i \cdot a}{f_i}$$

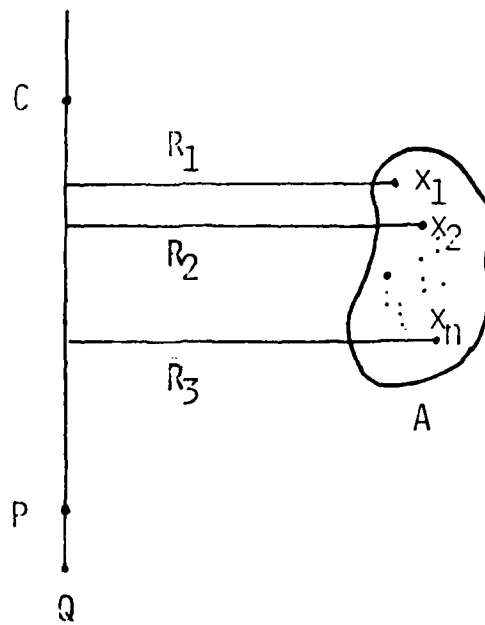


Fig. 1

AN OBJECT AND ITS DISTANCES FROM A REFERENCE PLANE

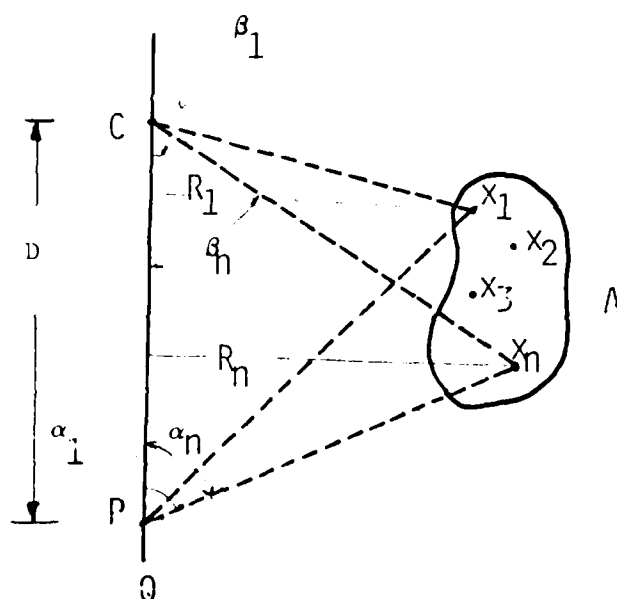


Fig. 2

TRIANGULATION GEOMETRY:  $R$  MAY BE FOUND IF  $\alpha, \beta$  AND  $D$   
ARE KNOWN

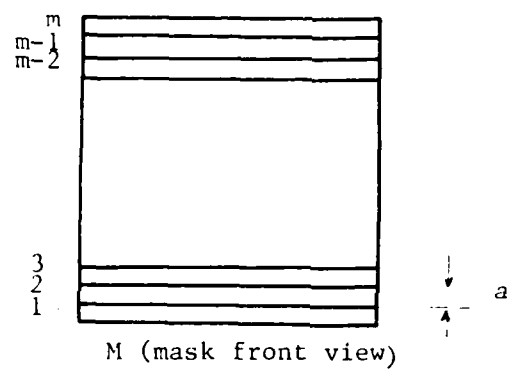


Fig. 3

MASK WITH  $m$  BARS EACH OF SIZE  $a$

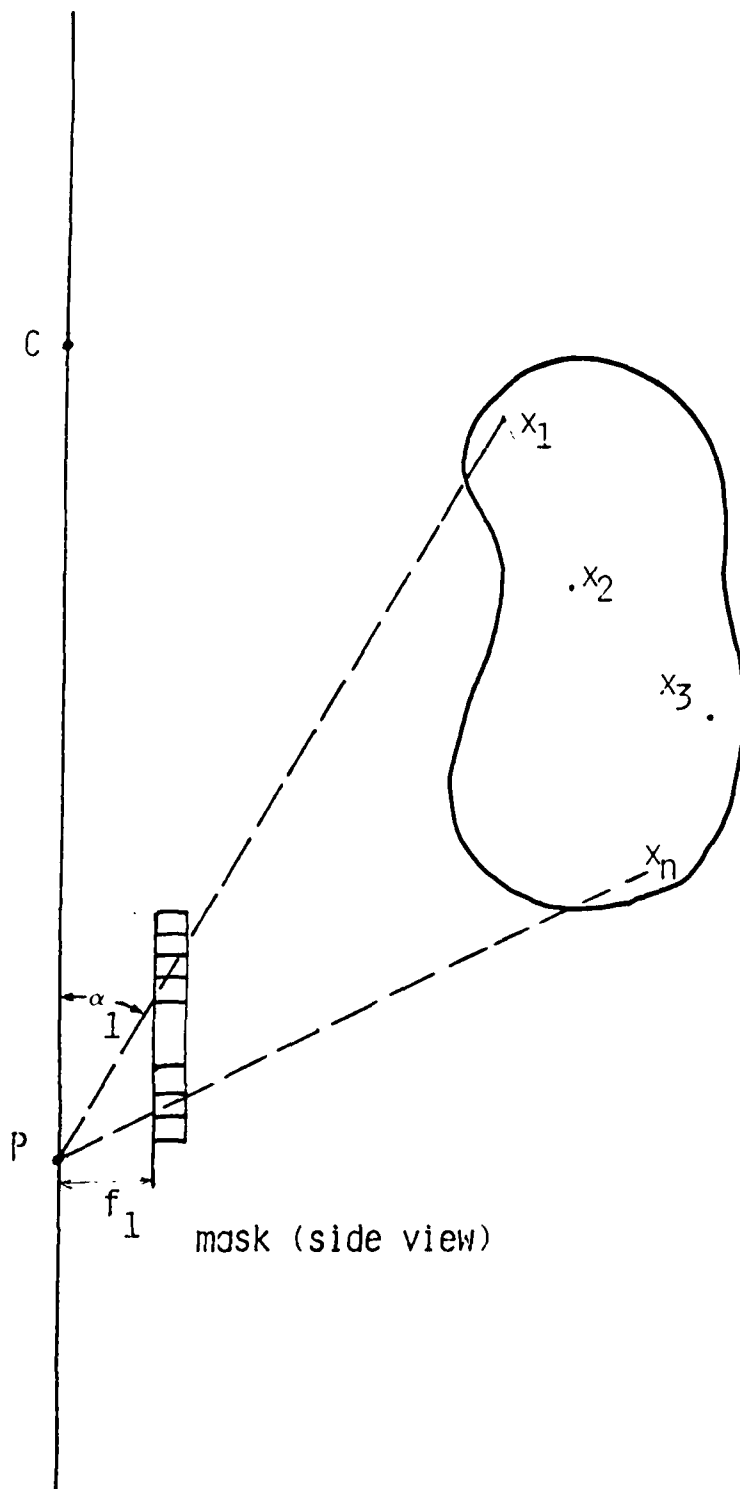


Fig. 4

USING THE MASK TO MEASURE  $\alpha$



where  $a$  is the height of each partition.

In the design and construction of this vision system,  $C$  is the focal point of a charge-injection device (CID) camera (General Electric Model TN 2500) with a  $244 \times 248$ -pixel resolution plane at focal distance  $f_2$  from  $C$ , the reference point  $P$  represents a high-intensity line source perpendicular to the plane of paper, and template  $M$  is made of glass so that the light from source  $P$  can pass through and illuminate object  $A$  (see Fig. 5).

It is very time consuming if we use a template such as that described above for measuring  $\cot \alpha_i$  of all points of the object  $A$ : the above design would fail to work in real time. Therefore, instead of template  $M$  we use a series of coded glass masks.

Figure 6 shows a reference mask "0" and eight masks numbered 1 through 8 which can be used for a  $(2^8 = 256) \times 256$  image.

In the following example we will show how such masks are used in finding  $\cot \alpha$ . For simplicity, however, we will use a  $4 \times 4$  image which implies  $\log_2 4 = 2$  masks. Note that in this example black has the highest gray value, "1", and white the lowest, "0". Also, the following criterion is exploited in comparing two images (one is called "reference" and the other "input" image): if the gray level of a pixel of an input image is equal to or less than the corresponding pixel in the reference image, a zero is stored in a memory corresponding to that pixel.

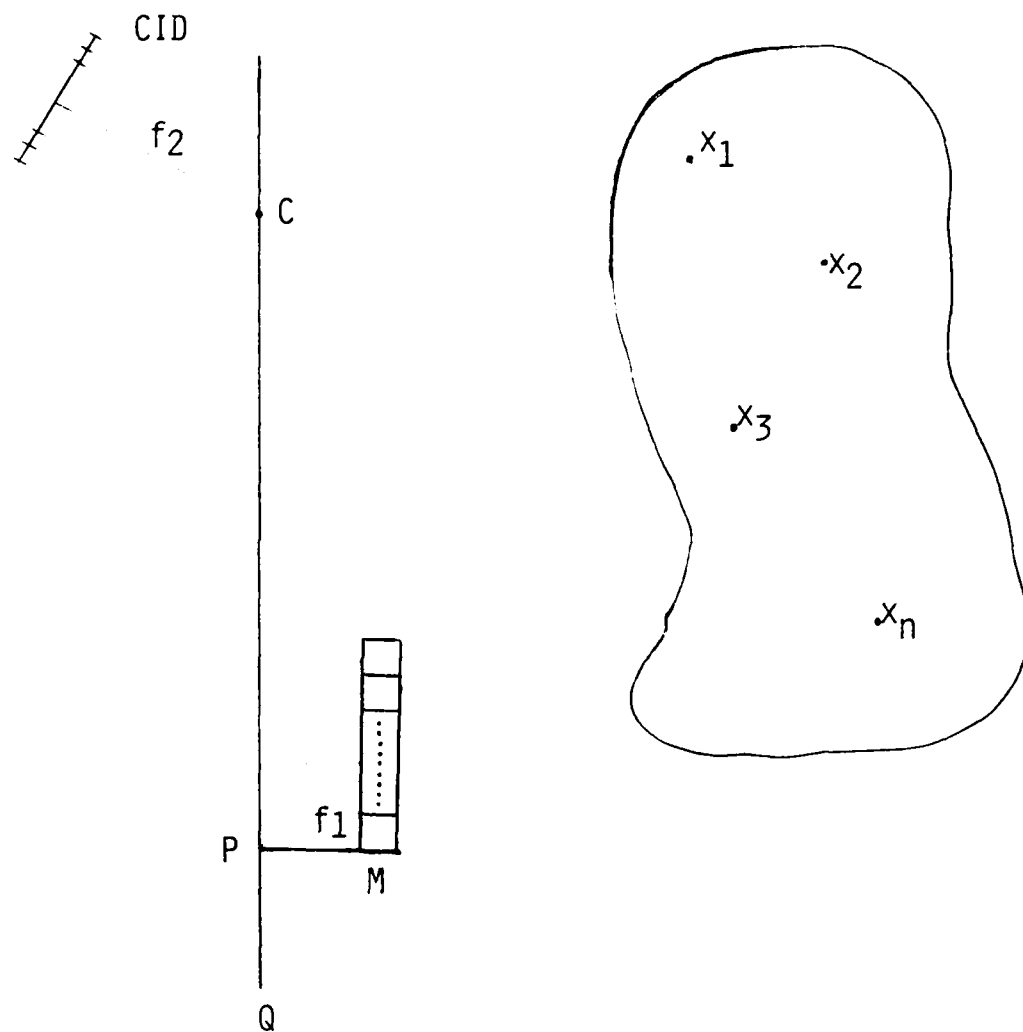


Fig. 5

A RANGING SYSTEM, WITH SOURCE, P, MASK M, AND CID CAMERA  
FOCUSED AT C

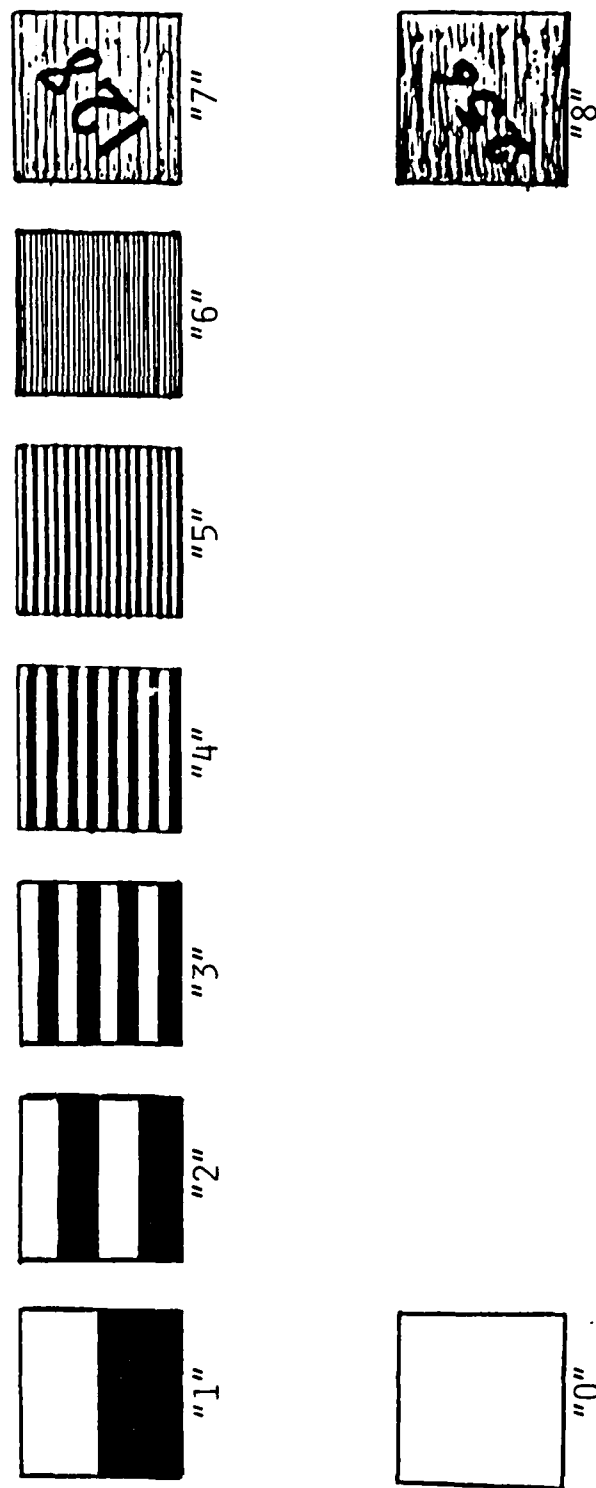


Fig. 6

MASKS OF ALTERNATING TRANSPARENT AND OPAQUE SECTIONS, AND THE REFERENCE MASK

Referring to Figure 7a, we note that the projector P illuminates the whole object A and the absolute gray values of various points of object A (e.g.  $x_1, x_2, x_3, x_4$ ) are stored in a memory plane called the reference memory after being imaged by the camera. This array is called the reference image.

In Figure 7b the first mask is placed in front of projector P. Since half of mask "1" is opaque, points  $x_3$  and  $x_4$  would not be illuminated; therefore, in comparing the resultant image with the reference image, the algorithm would give points  $x_1$  and  $x_2$  the same gray level, while  $x_3$  and  $x_4$  have a higher gray value. Thus the values "0" for former points and "1" for later points, respectively, will be stored in a memory called "Bit plane memory 1". Note that the values stored in bit plane memories are relative single-bit values for each pixel.

If we proceed in the same manner with mask "2" we will have results such as the ones in bit plane memory-2 of Figure 7c.

In the next step the corresponding values of comparison for each pixel will be collected, in sequential order, as a code (for our example with  $4 \times 4$  image we have 2 bit planes and hence 2-bit code for each pixel - see Figure 7d). Converting this code to decimal will tell us which partition  $j$  illuminated a point  $x_i$ , and from the previous discussion we can find  $\cot \alpha$ .

Note that, depending on the type of image we have (e.g.  $4 \times 4$ ,  $8 \times 8, \dots, 2^n \times 2^n$ ), we need 2, 3, ...,  $n$  masks in order to have an  $n$ -bit code that defines every point of an image.

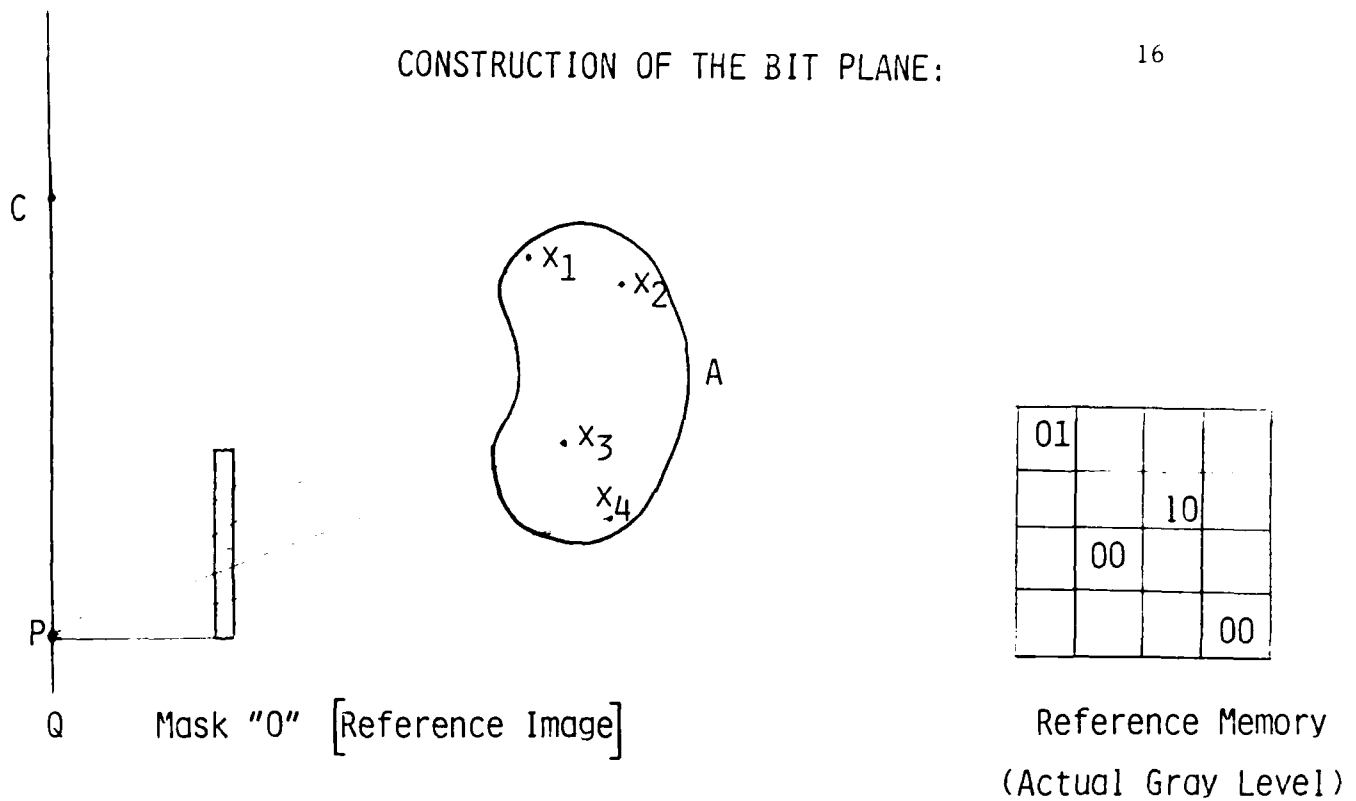


Fig. 7(a)

THE REFERENCE IMAGE, TO ACCOUNT FOR INCIDENT INTENSITY  
AND OBJECT REFLECTIVITY

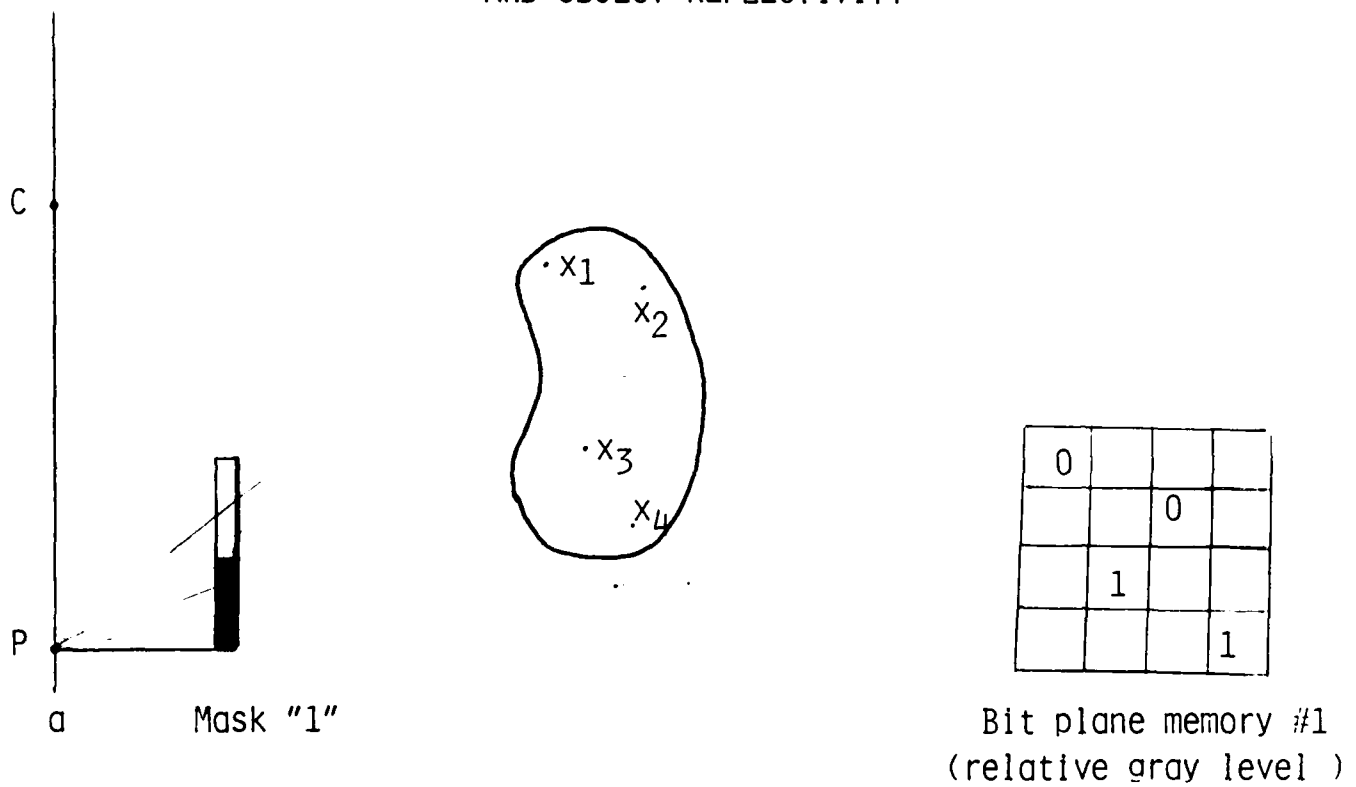


Fig. 7(b)

BIT-PLANE LEVEL ONE; RESULT OF TWO-BAR MASK

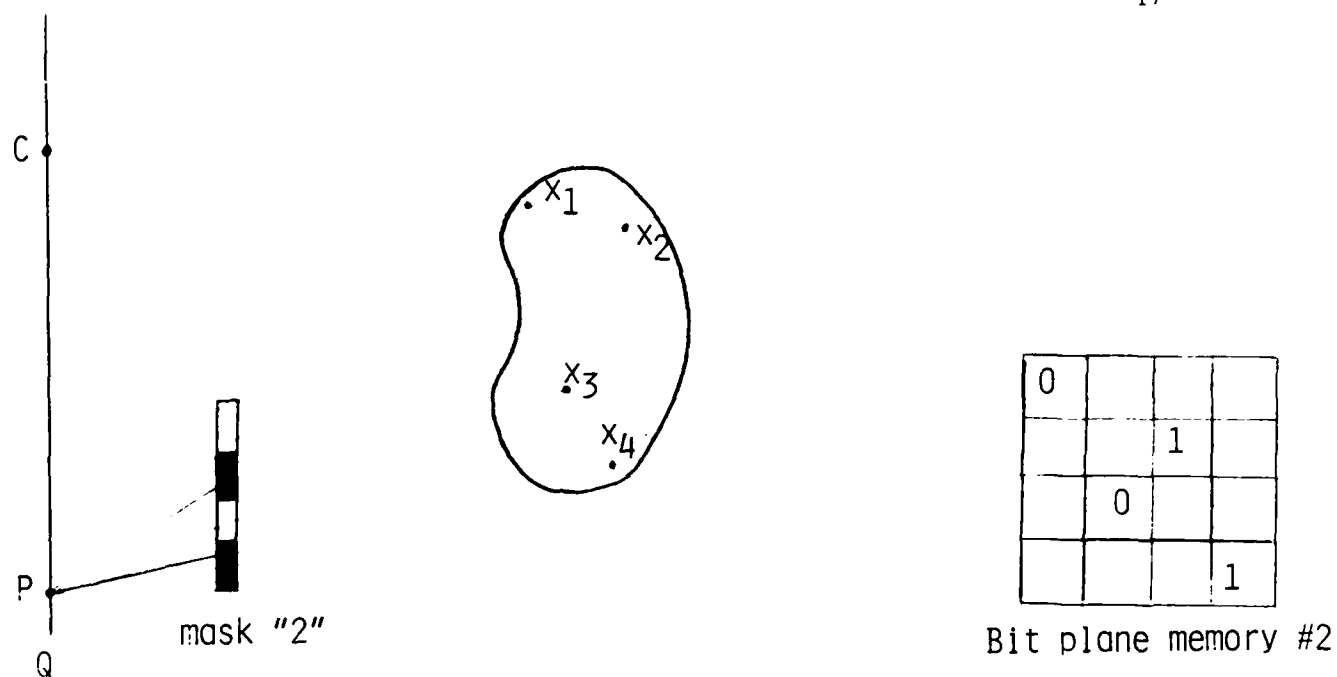


Fig. 7(c)

BIT PLANE LEVEL TWO; RESULT OF FOUR-BAR MASK

From: Bit plane #1  
 Bit plane #2

|     |     |     |     |
|-----|-----|-----|-----|
| 0 0 |     |     |     |
|     |     | 0 1 |     |
|     | 1 0 |     |     |
|     |     |     | 1 1 |

Resultant

Bit Plane

Fig. 7(d)

COLLECTING MASKS' RESULTS INTO A SINGLE CODE

Now that we know how to measure  $\cot \alpha$ , we need to know the value of  $\cot \beta$ . Referring to Figure 8, we suppose we have a camera with  $/x/$  CID plane (usually  $256 \times 256$ ), focal length  $f_2$ , and pixel size  $b \times b$ . Knowing which pixel we are looking at, we can calculate  $\cot \beta_i$  from the following relationship:

$$\cot \beta_i = \frac{(n-i) \cdot b}{f_2}$$

According to the above, if we have two look-up tables, one loaded with the values of  $\cot \beta_i$ , and the other one with the values of  $\cot \alpha_i$  calculated from data extracted from the resultant bit plane of Figure 7d, we can find the value corresponding to  $R_i$ .

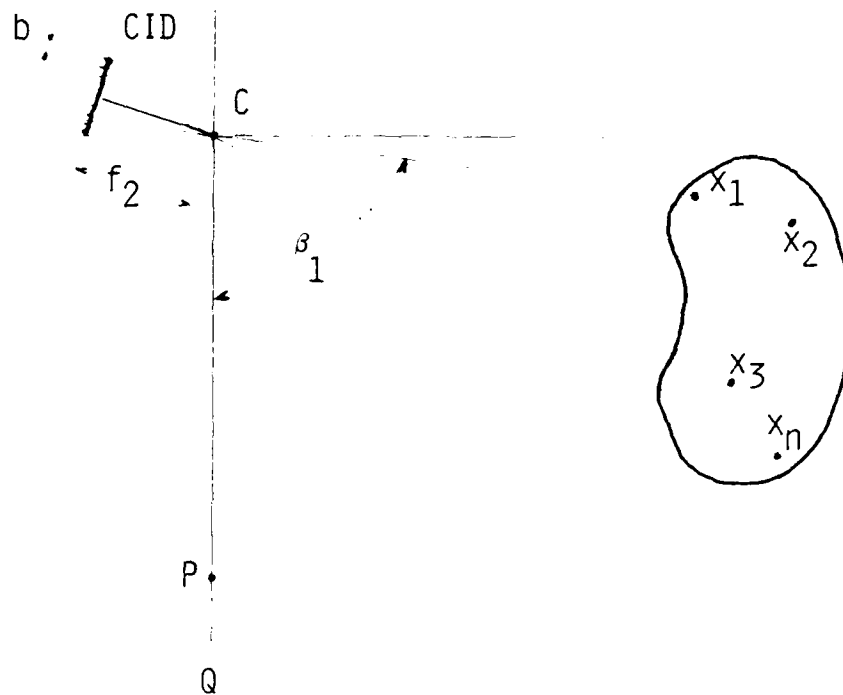


Fig. 8  
USING THE CAMERA TO MEASURE  $\beta$



## B.2 'Visible' field of View

By 'visible' field of view or 'visible' range we mean the area which is both illuminated by the projector and within the optical field of the camera. In this section we define the boundaries of this field of view and find the resolution limits for the CID camera.

From Fig. 9 we can write the equations for lines  $\bar{A}$ ,  $\bar{B}$ ,  $\bar{C}$  and  $\bar{D}$  which constitute the contour of the visible field.

$$\bar{A}: y_A = (d - x_A) \cdot \cot 2\theta$$

$$\bar{B}: x_B = 0$$

$$\bar{C}: y_C = x_C \cdot \frac{f_1}{m} - f_1$$

$$\bar{D}: x_D = d$$

Note:  $\bar{P}$  is at  
 $x=0, y = -f_1,$

$\bar{C}$  is at  $x=d, y=0,$   
and  $\theta$  is the half-angle of  
the field-of-view of the  
camera

The intersection of lines  $\bar{A}$  and  $\bar{C}$  gives the x-y coordinate (Y is actually the range) of the closest point O to the reference plane which is visible.

$$x_0 = \frac{d \cot 2\theta + f_1}{\frac{f_1}{m} + \cot 2\theta}$$

$$y_0 = \frac{\cot 2\theta (d-m)}{1 + \frac{m}{f_1} \cot 2\theta}$$

Intersection of  $\bar{C}$  and  $\bar{D}$ ,  $\bar{A}$  and  $\bar{B}$  will yield the locations of the other two closest visible points, one on the projector horizontal axis (Q) and the other on the horizontal axis and at the same height as the camera (E).

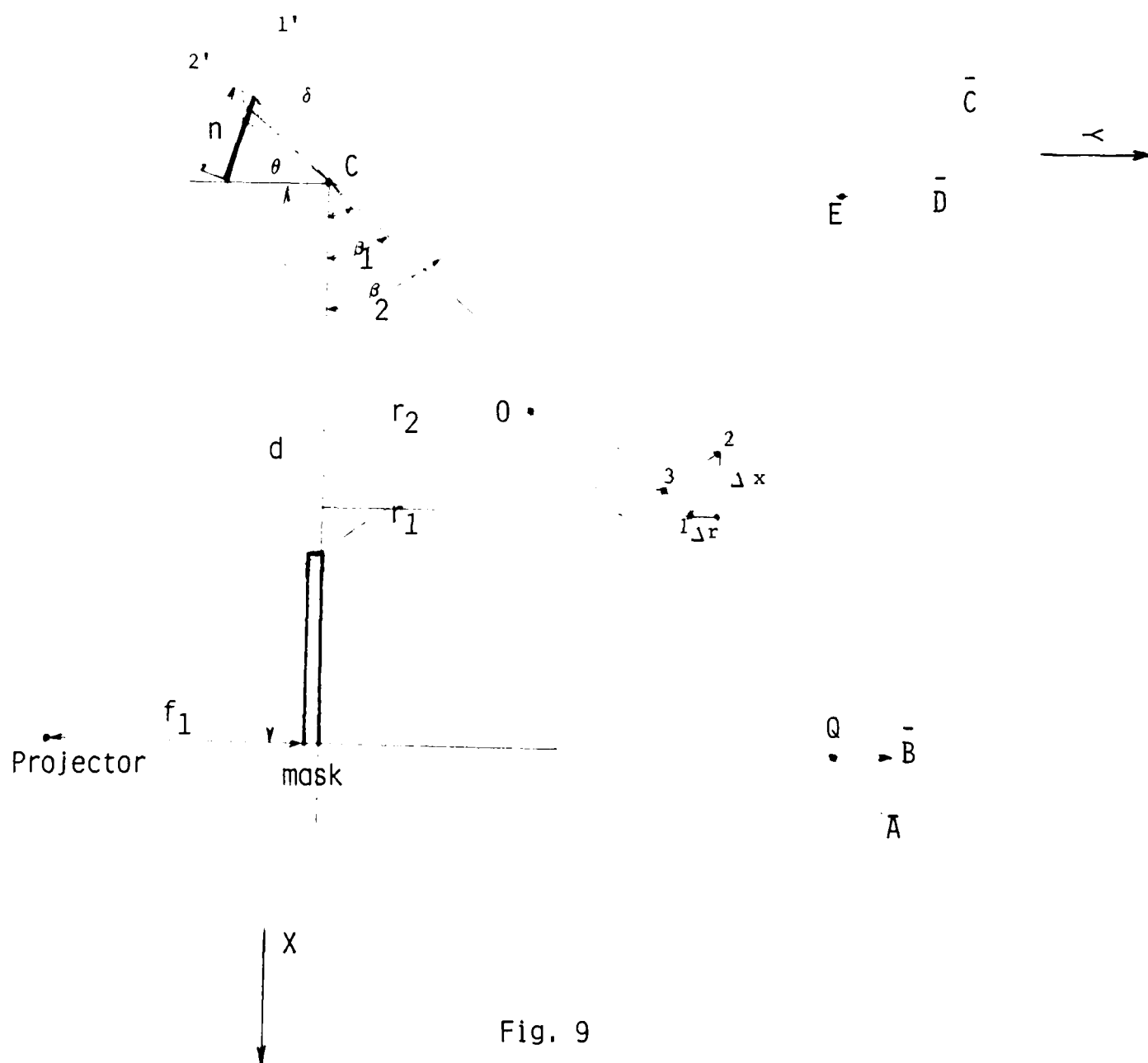


Fig. 9

DEFINING THE VISIBLE FIELD AND GEOMETRY OF THE  
DEFINITION OF RESOLUTION

That is,

$$x_Q = 0$$

$$y_Q = d \cot 2\theta$$

$$x_E = +d$$

$$y_E = d \cdot \frac{f_1}{m} - f_1$$

So the following result expresses the visibility of an object or a point on the object.

$$R_x = y_x \begin{cases} \geq + x \cdot \frac{f_1}{m} - f_1, & \text{for } x_E \geq x \geq x_0 \\ \geq (d - x) \cot 2\theta, & x_0 \geq x \geq 0 \\ \text{Invisible,} & \text{otherwise} \end{cases}$$

Therefore for a point of an object to be in the 'visible' field of view its range must be greater than or equal to  $f_1(\frac{x}{m} - 1)$   $x_0 \leq x \leq x_E$ , or greater than or equal to  $(d-x)\cot 2\theta$  if  $0 \leq x \leq x_0$ , where all the parameters are as defined previously.

Now we want to find the relationship between the resolution and the range: suppose from a point  $(x_1, r_1)$  we want to view another point  $(x_2, r_2)$  where:

$$x_2 = x_1 + \Delta x$$

$$r_2 = r_1 + \Delta r$$

Then,

$$\theta_1 = \tan^{-1} \frac{r_1}{d - x_1}$$

$$\beta_2 = \tan^{-1} \frac{r_1 + \Delta r}{d - (x_1 + \Delta x)}$$

and

$$\alpha_1 = 90 - (\theta + \beta_1)$$

$$\alpha_2 = 90 - (\theta + \beta_2)$$

Now the resulting images will be:

$$1' = \frac{n}{2} \pm \left[ \frac{n}{2} \cot \theta_1 \cdot \tan \alpha_1 \right]$$

(if  $\beta$  is more than  $(90 - \theta)$  the  
plus sign must be used)

$$2' = \frac{n}{2} \pm \left[ \frac{n}{2} \cot \theta_2 \cdot \tan \alpha_2 \right]$$

and the difference (or displacement) of the image by moving from point 1 to point 2 is:

$$\Delta R' = |1' - 2'|$$

Therefore to be able to distinguish between different points we require

$$\begin{array}{c} \text{Pixel} \\ \text{Size} \end{array} \leq \Delta R' \leq n - (\min \{1' \text{ and } 2'\})$$

In order to be able to resolve a point as a function of its previous location (range and height) its image must differ by value greater than the pixel size of the camera (otherwise both points would be resolved as one) and less than  $n$  (the width of CID plane) minus minimum of the image value of either points 1 and 2 (namely  $1'$  and  $2'$ ). (See Appendices A and B for tabulation and display of typical values of visible ranges and precision of range estimates.)

### B.3 Camera-Projector Set Up

To calculate the range it is necessary to arrange the apparatus as illustrated in Figure 10. Since  $\alpha_1$  or  $\cot \alpha_1$  is to be decoded using the masks, the transparency of the stripes in the masks must vary in the Z direction. If, however, the set-up is like that of Figure 11, the variation in the masks is in the Y direction. That results in decoding  $\alpha_2$  or  $\cot \alpha_2$ , which do not contain any range information.

### B.4 Light Source and Slit

An important consideration in this system is the necessity of having a fine light source with the narrowest possible slit while providing the required light intensity. The need for a narrow slit is illustrated in Fig. 12.

As can be seen from Figure 12a the shaded areas are neither completely illuminated nor completely dark. Therefore, in comparison with the reference memory these areas will result in wrong encodings. On the other hand, Figure 12b shows how a perfect (but non-realizable) line slit will result in just two distinct areas: illuminated and dark.

Therefore one must approximate a perfect slit. However, with the light source that we had available (Sunpak 433D) it was not possible to provide enough light through the slit; the reduced intensity resulted in degradation of images and hence of the estimates of range.

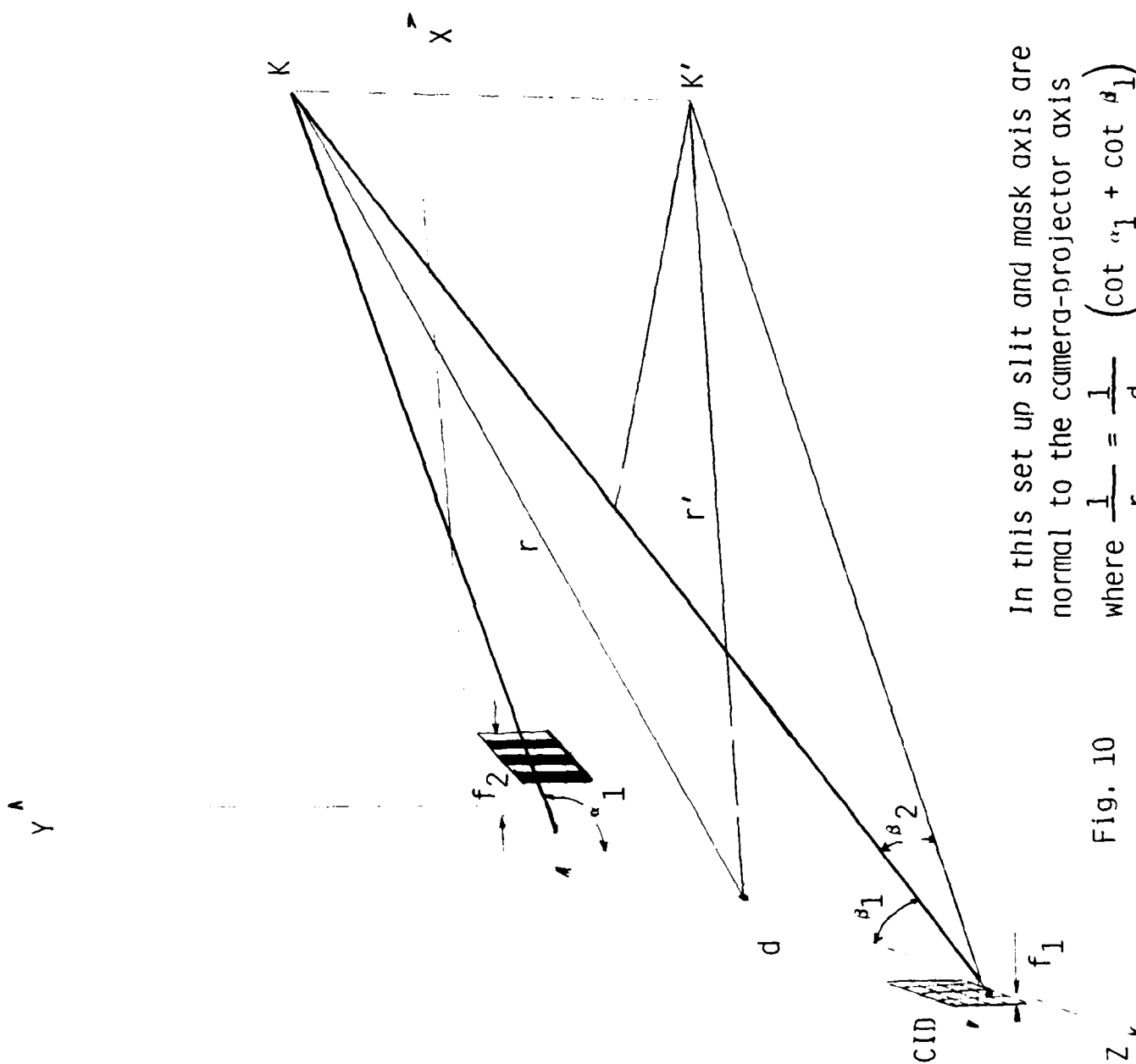


Fig. 10

In this set up slit and mask axis are normal to the camera-projector axis where  $\frac{1}{r} = \frac{1}{d} (\cot \alpha_1 + \cot \beta_1)$

CORRECT ORIENTATION OF APPARATUS FOR MEASURING RANGE

AN INCORRECT ORIENTATION;  $\alpha_2$  WILL BE MEASURED,  
WHICH DOES NOT CONTRIBUTE TO RANGE INFORMATION

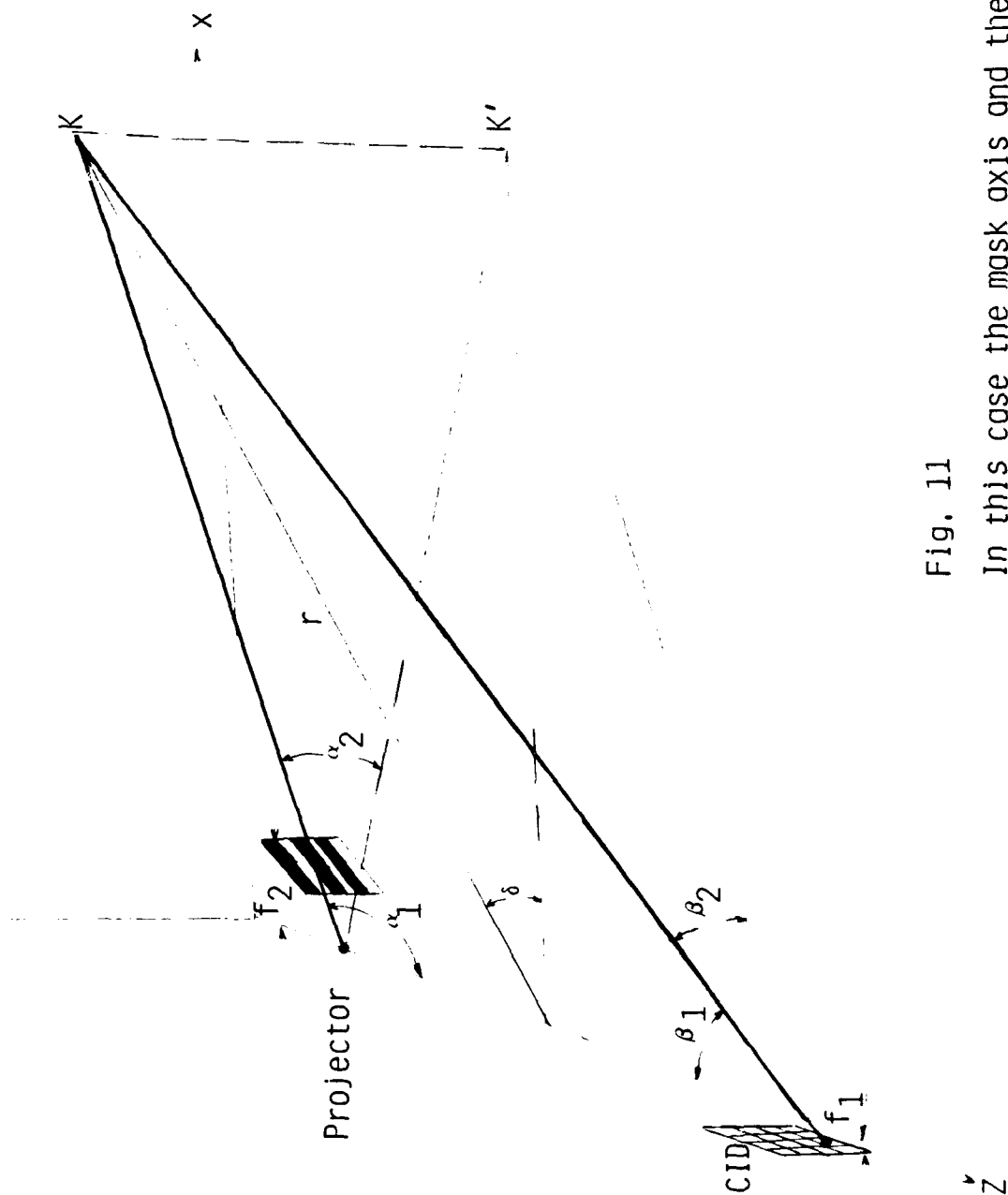
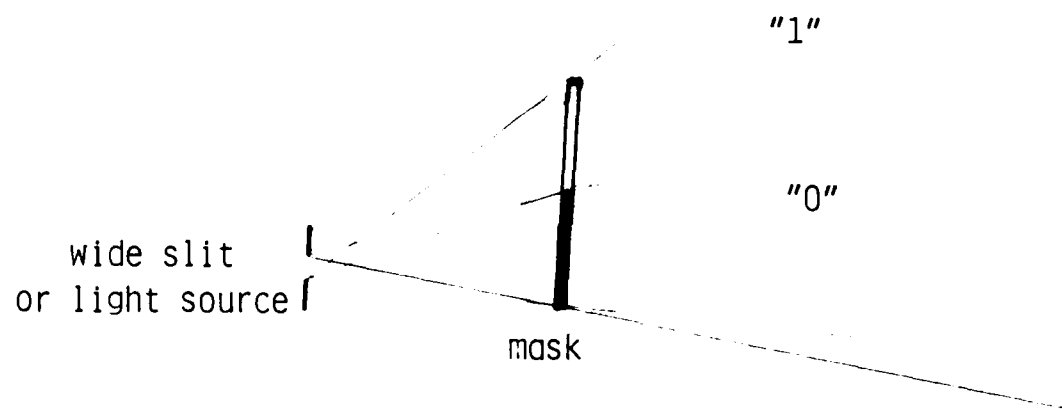


Fig. 11

In this case the mask axis and the  
projector-camera axis are parallel

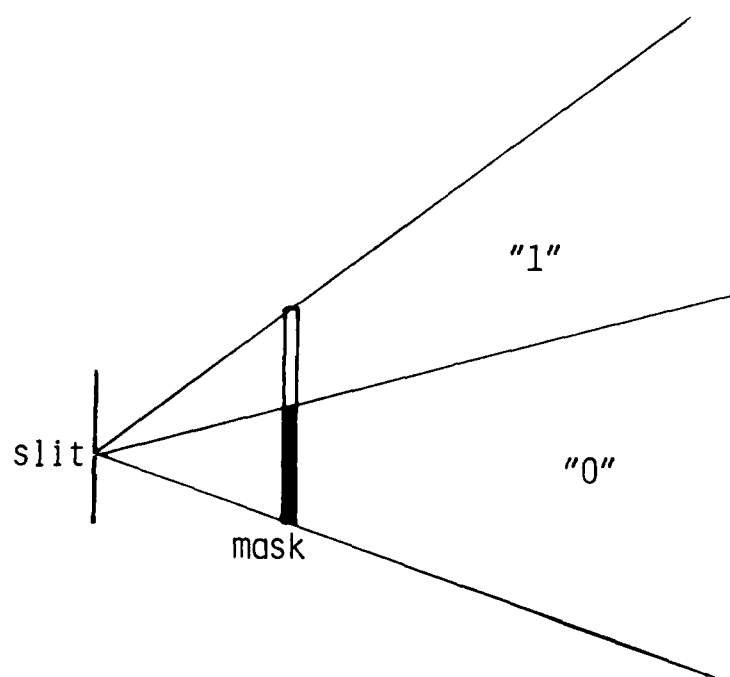


a) wide slit

Fig. 12

A WIDE SLIT YIELDS AMBIGUOUS REGIONS





b) line slit (perfect slit)

Fig. 12

A PERFECT SLIT YIELDS ONLY DICHOTOMY

To compensate for this problem we had to modify the configuration of Figure 12a, by using an adjustable slit (see Fig. 13). We could adjust the width of the slit to the smallest that would give the sharpest possible images.

As noted above, the use of a non-perfect slit would degrade the range result, especially through the finest masks since the width of the slit would be almost the same as that of the smallest mask division. In the following paragraph this degradation is calculated. It should be pointed out that if the slit causes degradation for a given mask, all the masks having finer divisions will be degraded as well. For example, if we cannot have a sharp image of the mask no. 6, masks no. 7 and 8 will also be degraded.

As explained in the algorithm description, there are eight masks (1-8) which will result in the constitution of an 8-bit code relating to  $\cot \alpha$ :

$$\cot \alpha \approx [\text{decimal } (B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7)] \times \text{constant } (= \frac{a}{f_1})$$

(where  $a$  = width of mask's bars, and  $f_1$  = distance from mask to slit)

Therefore, if mask  $i$  ( $i=0, \dots, 7$ ) is degraded (i.e., gives bad image) the resulting code will have degradation in  $2^{8-i}$  bars of the finest mask. For example if the last mask ( $i=7$ ) does not give a good image, the code can be  $B_0 B_1 B_2 \dots B_6 0$ , or  $B_0 B_1 B_2 \dots B_6 1$  which means we cannot distinguish between the  $n$ -th bar and the  $(n+1)$ -th bar of the mask; in turn, that will reduce the range precision. Suppose  $A$  is the point for which we are

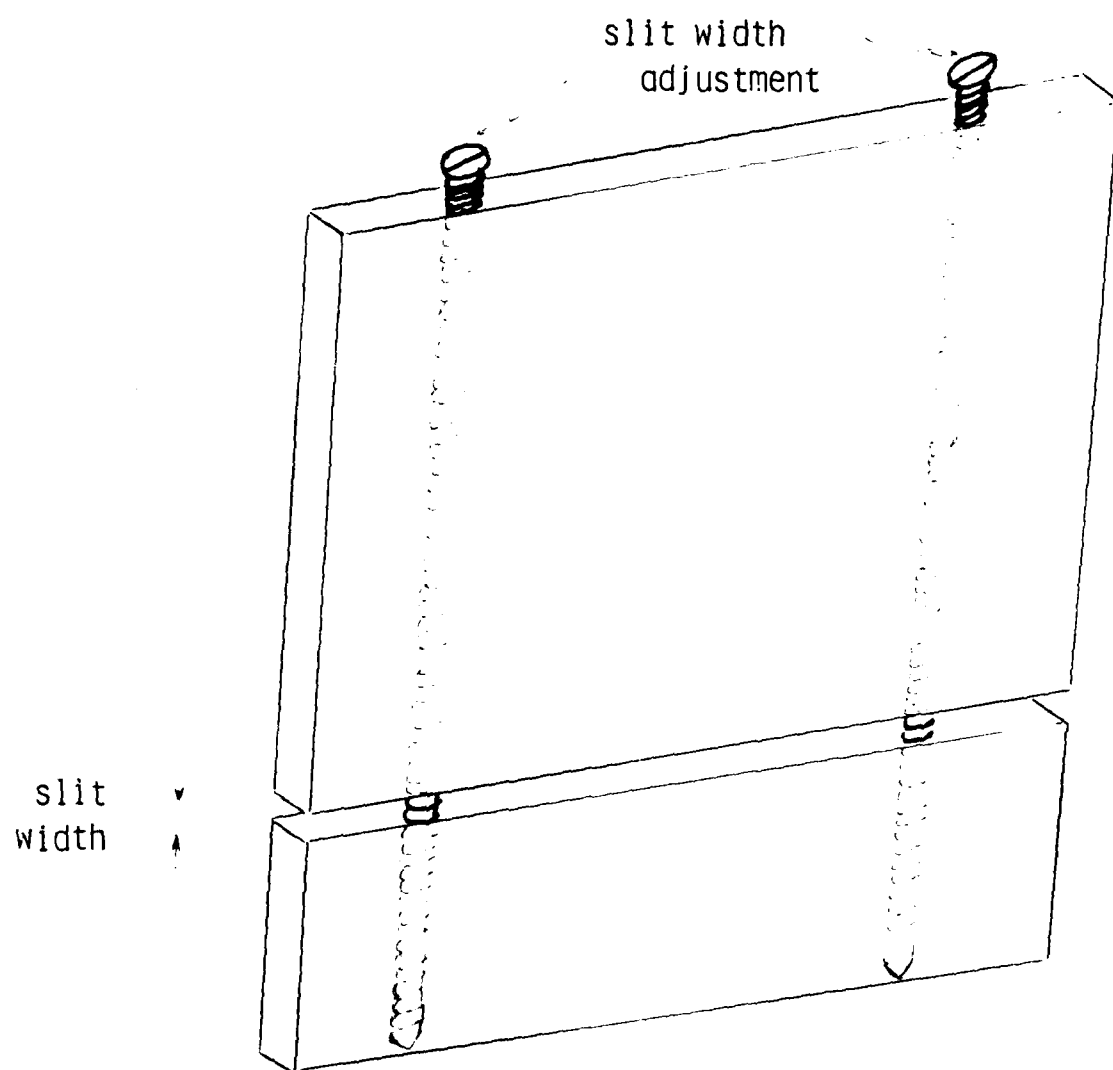


Fig. 13

ADJUSTABLE SLIT USED IN THE RANGING SYSTEM

calculating the range. Let the  $n$ -th bar be the one illuminating point A. Therefore,

$$\cot \alpha_1 = n \cdot \frac{a}{f_1}$$

Now if we have image degradation for mask  $i$ , we might decode  $(n \pm 2^{8-i})$  for  $\cot \alpha$  which results (see Fig. 14) in

$$\cot \alpha_2 = (n \pm 2^{8-i}) \cdot \frac{a}{f_1}$$

and

$$R = \frac{d}{\cot \alpha + \cot \beta}$$

$$\text{Let } \cot \beta = m \cdot \frac{b}{f_2} = m \cdot \frac{a}{f_1}$$

therefore

$$R' = \frac{d}{(n + m \pm 2^{8-i}) \cdot \frac{a}{f_1}}$$

$$R = \frac{d}{(n + m) \cdot \frac{a}{f_1}}$$

$$\frac{R}{R'} = \frac{n + m \pm 2^{8-i}}{n + m} \quad \therefore \frac{R}{R'} = 1 \pm \frac{2^{8-i}}{n + m}$$

Now if we consider the 'gray area' created by the width of the slit, from the following drawing we can see that where we have a black stripe the gray field does not affect our result since the gray level of 'gray area' is smaller than black therefore a 'zero' will be stored by the comparator, whereas in

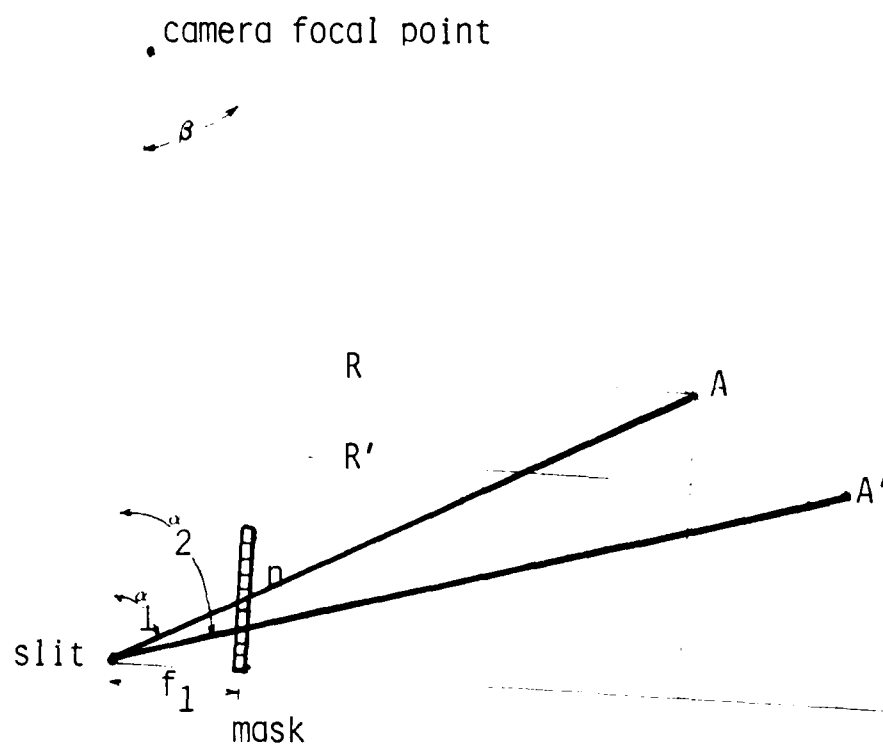


Fig. 14

GEOMETRY FOR EFFECT OF GRAY-LEVEL UNCERTAINTY  
ON RANGE RESOLUTION

white stripe the gray field will affect the true comparison. It is for this reason that we use the 'false-black' as a more proper name for 'gray area'.

In the following paragraphs the ratio of false black area to correct detected area is computed.

Fig. 15 shows the effect of false-black (gray-area) on the accuracy of determining range of an object. It will be shown that the false-black area caused by a stripe (white or clear stripe) of the mask depends only on the number  $n$  of white-stripe to black-stripe transitions, the width of the slit  $l$ , the projector-mask distance  $f_1$ , and the range of the object  $R$ .

Referring to similar triangles ABC and A'B'C' we have:

$$A'B' = l \cdot \frac{R - f_1}{f_1}$$

also in triangles ABD and A''B''D we can write:

$$A''B'' = l \cdot \frac{R - f_1}{f_1}$$

Therefore:

$$\begin{aligned} A'B' &= A''B'' = \text{gray area of an alternation of stripes} \\ &= \left( \frac{R}{f_1} - 1 \right) l \end{aligned}$$

Hence, for a mask containing  $n$  alternations, we have:

$$\text{Total false-black area} = n \left( \frac{R}{f_1} - 1 \right) l$$

The ratio of false-detected area to correctly-detected area due to the false-black problem is therefore:

$$E_n = \frac{\text{total false-black area}}{(\text{detectable-area}) - (\text{false-detected area})}$$

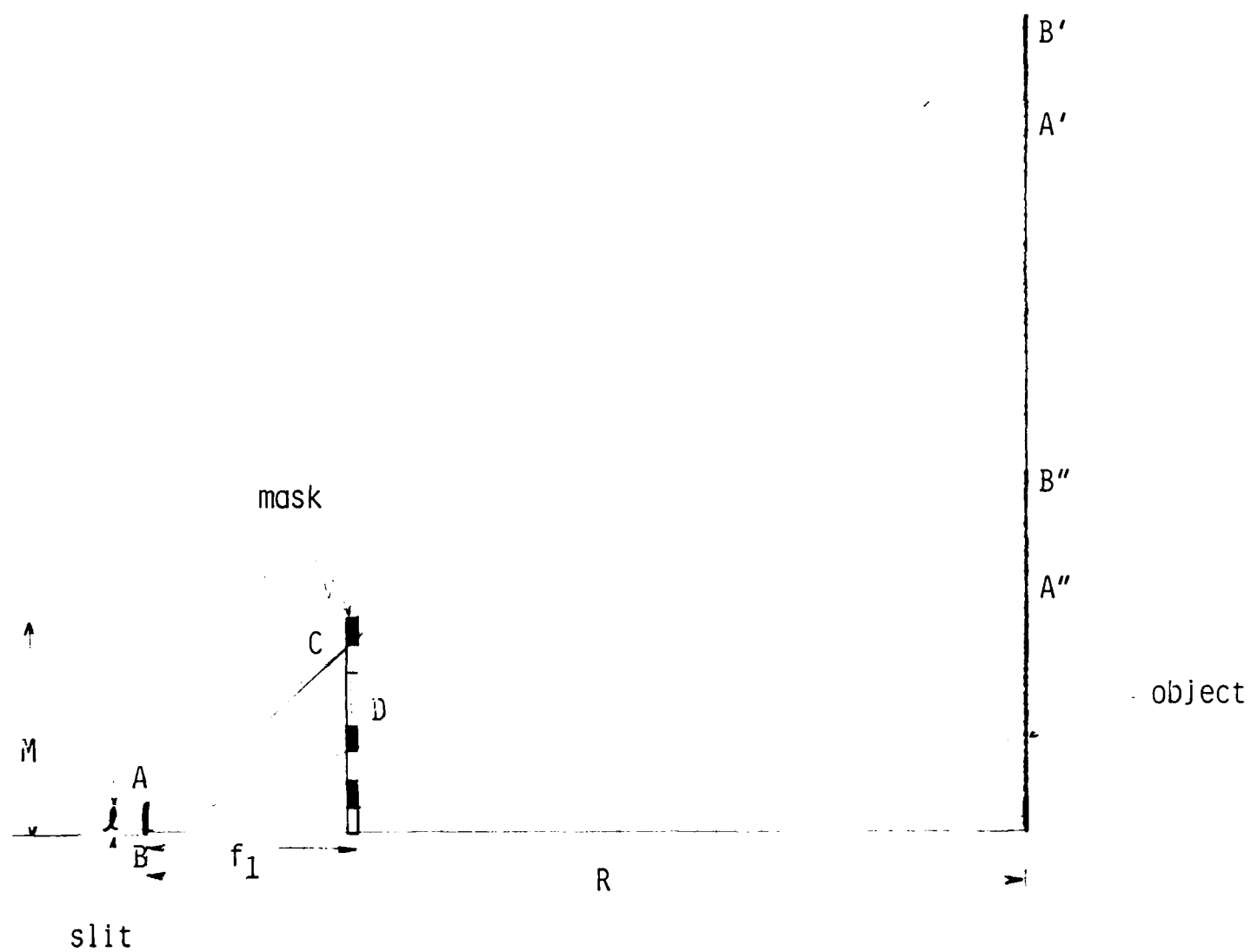


Fig. 15

GEOMETRY FOR CALCULATION OF TOTAL FALSE-BLACK AREA

$$= \frac{n \left( \frac{R}{f_1} - 1 \right) I}{M \cdot \frac{R}{f_1} - n \left( \frac{R}{f_1} - 1 \right) I}$$

$$E_n = \frac{n (R - f_1) I}{MR}$$

where  $M$  is the mask width.

#### B.5 Hardware-Algorithm Modification

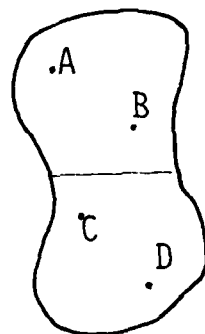
The DAD box compares the live video input with the output of the reference memory (on a pixel-by-pixel basis); if the input pixel is less than or equal to the reference pixel, a zero is stored. If it is greater, a one is stored.

However, considering 'white' the highest gray level and 'black' the lowest gray level, physically it is not possible to have live inputs with gray level higher than reference memory. Therefore, we had to invert the contents of reference memory. This could be done using TTL inverters (Table 1). However, it was easier to invert any input to the DAD box.

Experiments with the modified system yielded results that were incorrect and unpredictable. Further detection showed that the Gray-code to binary-code converter was causing this randomness and unpredictability because inversion of the reference memory which in turn would invert the 8 bit code of 8 bit-plane memories (i.e. instead of 1101110 the code 0010001 will be decoded). This resulted in the need for another inversion right before bit-plane inputs (Figure 16). The following state tables illustrate the above reasoning:

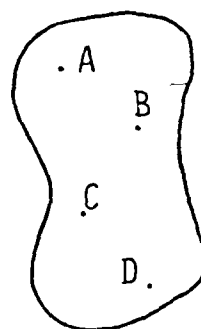


| object | Ref. | Input | Bit-plane(B1) |
|--------|------|-------|---------------|
| A      | 0    | 1     | 1             |
| B      | 0    | 1     | 1             |
| C      | 0    | 0     | 0             |
| D      | 0    | 0     | 0             |



1st video input

| object | Ref. | Input | B2 |
|--------|------|-------|----|
| A      | 0    | 1     | 1  |
| B      | 0    | 0     | 0  |
| C      | 0    | 0     | 0  |
| D      | 0    | 1     | 1  |



2nd video input

| object | binary | equiv. gray | $B_1B_2$ |
|--------|--------|-------------|----------|
| A      | 00     | 00          | 11       |
| B      | 01     | 01          | 10       |
| C      | 10     | 11          | 00       |
| D      | 11     | 10          | 01       |

As we can see  $B_1B_2$  is exactly inverse of true gray code.

Fig. 16

CONSTRUCTION OF CORRECT CODES USING INVERSION

| Ref. | Input | Bitplane     | Ref. | Input | Bitplane    |
|------|-------|--------------|------|-------|-------------|
| 0    | 0     | 0            | 1    | 0     | 0           |
| 0    | 1     | not possible | 1    | 1     | not allowed |
| 1    | 0     | 0            | 0    | 0     | 0           |
| 1    | 1     | 0            | 0    | 1     | 1           |

we have no state '1'

now we have permissible state '1'

Table 1

### B.6 Look-up Tables' Restriction of Resolution and Dynamic Range

The mapping RAMs (look-up tables [L.U.T.s]) are two separate 256 8-bit word memories for storage of the values of  $\cot \alpha$  and  $\cot \beta$ . Values from these two L.U.T.s are added using an eight-bit A.L.U. Therefore the following considerations apply:

- a) Since the mapping RAMs are eight-bit, we are allowed to have values just from 0 to 255. The largest dynamic range will be achieved when

$$\frac{a}{f_1} = \frac{b}{f_2}, \text{ where}$$

$a$  is the finest mask division

$b$  is the pixel width

$f_1$  = projector-mask distance, and

$f_2$  = camera focal length

- b) Since the A.L.U. output is eight bits we cannot use the full dynamic range of the L.U.Ts. In order to avoid overflow we have to use half of dynamic range (0-127) of

the L.U.T. This in turn implies the following resolution problem: since the addressing of these memories is by autoincrement and the readout of L.U.T. proceeds from 0th to 255th location, and the corresponding addition in A.L.U. also takes place under the same criteria, we have to load all the locations (256) of look-up tables with 128 (0-128) data. Therefore, every two consecutive locations were loaded with the same value. It is shown in Fig. 8 how the resolution is affected.

$$\text{Now, } \cot \beta = \frac{b}{f_2}$$

$$\text{and since } \frac{a}{f_1} = \frac{b}{f_2} \therefore \cot \beta_{\text{unit}} = \cot \alpha_{\text{unit}}$$

where  $\alpha_{\text{unit}}$  is the illumination angle, with respect to the reference plane, as if the object (or a point on the object) were illuminated through the first stripe of the mask, and  $\beta_{\text{unit}}$  is the imaging angle, with respect to the reference plane, as if the image of the object were received by the pixel next to the center of CID array.

Now,

$$\cot \alpha_{\text{unit}} = \frac{b}{f_1}$$

$$r_A = n \cot \alpha_{\text{unit}} + (m+1) \cot \beta_{\text{unit}} = (n+m+1) \cot \alpha_{\text{unit}}$$

$$r_B = (n+1) \cot \alpha_{\text{unit}} + (m+1) \cot \beta_{\text{unit}} = (n+m+2) \cot \alpha_{\text{unit}}$$

$$r_C = (n+1) \cot \alpha_{\text{unit}} + m \cot \beta_{\text{unit}} = (n+m+1) \cot \alpha_{\text{unit}}$$

$$r_D = (n) \cot \alpha_{\text{unit}} + m \cot \beta_{\text{unit}} = (n+m) \cot \alpha_{\text{unit}}$$

According to the preceding explanation, the contents of  $m$  and  $m+1$ , and  $n$  and  $n+1$  in the look-up tables are equal respectively; therefore, four points A, B, C, and D will be determined as being in the same range from 'reference plane.'

The resolution is affected as a function of range as follows: (Figure 18 is just part of Figure 17) as we can see, the resolution depends on both  $R$  and the lateral location of the object.

$$h_A = R \cot \alpha$$

$$\frac{h_A}{R} = \frac{n \cdot a}{f_1}$$

$$\text{since right sides are equal } \frac{h_A}{R} = \frac{y_D}{R + \Delta R}$$

$$\frac{h_D}{R + \Delta R} = \frac{n \cdot a}{f_1}$$

$$\frac{h_A}{h_D} = \frac{R}{R + \Delta R} \therefore \frac{h_D}{h_A} = 1 + \frac{\Delta R}{R} \rightarrow \pm \left( \frac{h_D}{h_A} - 1 \right) R$$

The other way to approach the resolution problem is to observe that since

$$\frac{1}{R} = \frac{1}{d} r \quad \therefore R = \frac{d}{r}$$

$$\Delta R = \pm d \left| \left( \frac{1}{(n+m+1)} - \frac{1}{n+m} \right) \tan \alpha_{\text{unit}} \right|$$

$$= \pm d \left| \left( \frac{n+m-n-m-1}{(n+m+1)(n+m)} \cdot \frac{a}{f_1} \right) \right|$$

$$\Delta R = \pm d \cdot \frac{a}{f_1} \cdot \frac{1}{(n+m+1)(n+m)}$$

in which  $\frac{1}{(n+m+1)(n+m)}$  is a function of  $h_D < h_A$  and  $R$ .

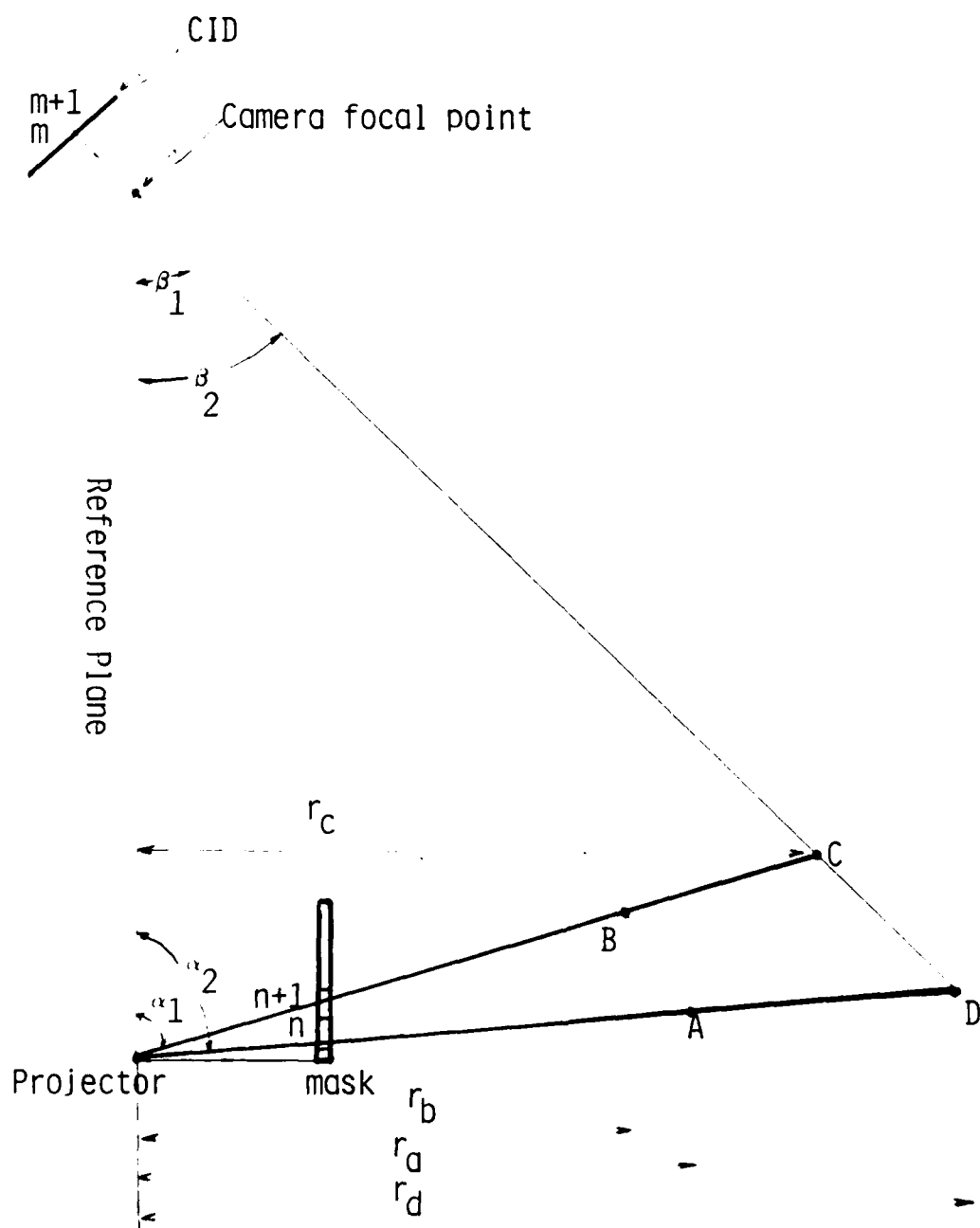


Fig. 17

GEOMETRY FOR EVALUATION OF INTRINSIC RANGE UNCERTAINTY



## ANOTHER VIEW OF RANGE UNCERTAINTY

## ANOTHER VIEW OF RANGE UNCERTAINTY

### B.7 Dark Objects' Range

One problem implicit in the method of implementing the algorithm is concerned with dark objects (including the whole object or some dark spots on the object with bright background) or holes in the object. The problem arises from the fact that whether it is a dark object or a hole, and whether the mask is covering this area or not, the result on the CID will be the same and therefore a zero will be stored in the bit-planes. Therefore the correct code will not be encoded for that area but will be stored as 0000000 = 0.

### B.8 Mask Adjustment

Another fact that one should carefully deal with is mask adjustment. The masks should not only be aligned with respect to each other but they must also be aligned with respect to the rotating drum on which they are mounted (i.e., parallel to the plane of rotation).

Adjusting the masks could be elaborate (such as using a computer and associating the video inputs with different masks). One simple procedure, however, is to use an overhead projector, project the masks on the board and adjust them by comparing the base lines, etc. (This is the method we used.) The problem that might occur with this approach is that if the first mask is not parallel to the rotation plane, then all the other masks will be tilted. To avoid this problem we must first level the projector parallel to the floor and then use the floor as a reference line.

### III. Topics investigated

#### A. Hardware

The design of the prototype structured-light source used a conventional high-fidelity turntable that had been modified by the erection of a metal wall about the circumference of (but not attached to) the platter. A hole (of the size and shape of one mask) was cut into the wall at the front of the device, and a top fabricated to confine the light so that it emerged only from the hole in front. A second wall was built on, and attached to the circumference of, the rotating platter of the turntable. A set of nine holes was cut into that wall for the insertion of the glass masks (eight bit-plane masks and one reference-image mask) described above. At the center of the interior of this circular section was the light source (the photographic flash unit), suspended from a bar attached to the top of the outer wall, and hence stationary.

The original goal was the real-time (30 frames per second) production and processing of range images; a 30 rotation-per-second rate of the turntable would have therefore been required. For reasons described below, processing at that rate was not possible, and the masks were rotated past the front opening by hand, at a very low speed.



The DAD box (so called because of its manufacturer, Digital-Analog Designs, Inc.) was designed to provide rapid storage of the eight bit-plane images, and their aggregation for definition of the eight-bit word needed to describe each pixel. The box (see technical description in Appendix C) also contained a fast arithmetic logic unit and the look-up tables described above. These latter two elements were intended to create range images rapidly, using the techniques described above. The DAD box was controlled by a Franklin 1000 microcomputer, for which software was written (see description and listings in Appendix D) to trigger and synchronize the light source. Although the software could, in principle, operate at a speed that would permit the 30-image-per-second rate, problems with the turntable speed control kept it turning too rapidly for the system. Accordingly, efforts were aimed at providing proof-of-principle rather than achieving maximum speed. The turntable was therefore rotated by hand, pausing at each of the nine masks to illuminate the object and record the result with the CID camera.

## B. Software - Range Data Recognition and Object Matching

### B.1. Introduction

This section describes a simulation system for range data recognition and the object matching. In fact, it is a range image processing algorithm which uses the data obtained by our computer vision system.

The simulation system began by generating the range data which are the distances from the sensor plane to the surfaces of a convex, plane-faced 3-D object (in a specific orientation). The range image processing program is composed of a range data recognition algorithm and a single view-to-3D object matching algorithm.

The goal of the range data recognition algorithm is to find

- (i) how many planes in the range image
- (ii) the plane equations
- (iii) the node coordinates of the object (visible in the single view).

The matching algorithm is used to prove whether this single view is one of the views of the reference object or not.

Because the range data image is a kind of intensity image, some of the mature techniques in intensity image processing were applied to our range image processing algorithm, including image segmentation, histogram smoothing and chain-code boundary representation. The simulation system achieved the desired results, but it is useful only for convex, plane-faced objects.

## B.2 Simulation System

Fig. 1 shows the simulation system we used. Plate 1 is a summary of the steps described below.

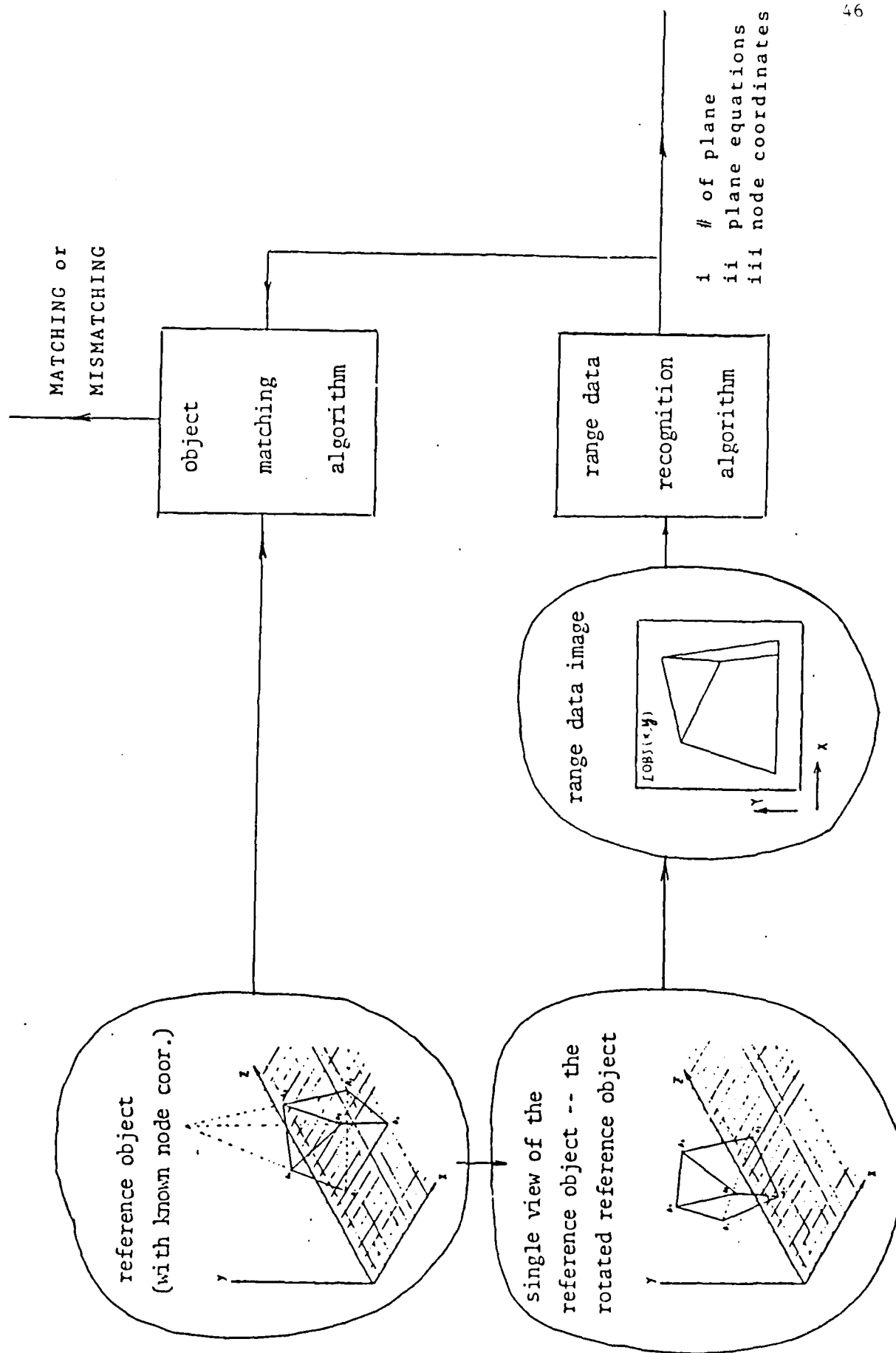


Fig. 1  
SUMMARY OF THE SIMULATION PROCEDURE

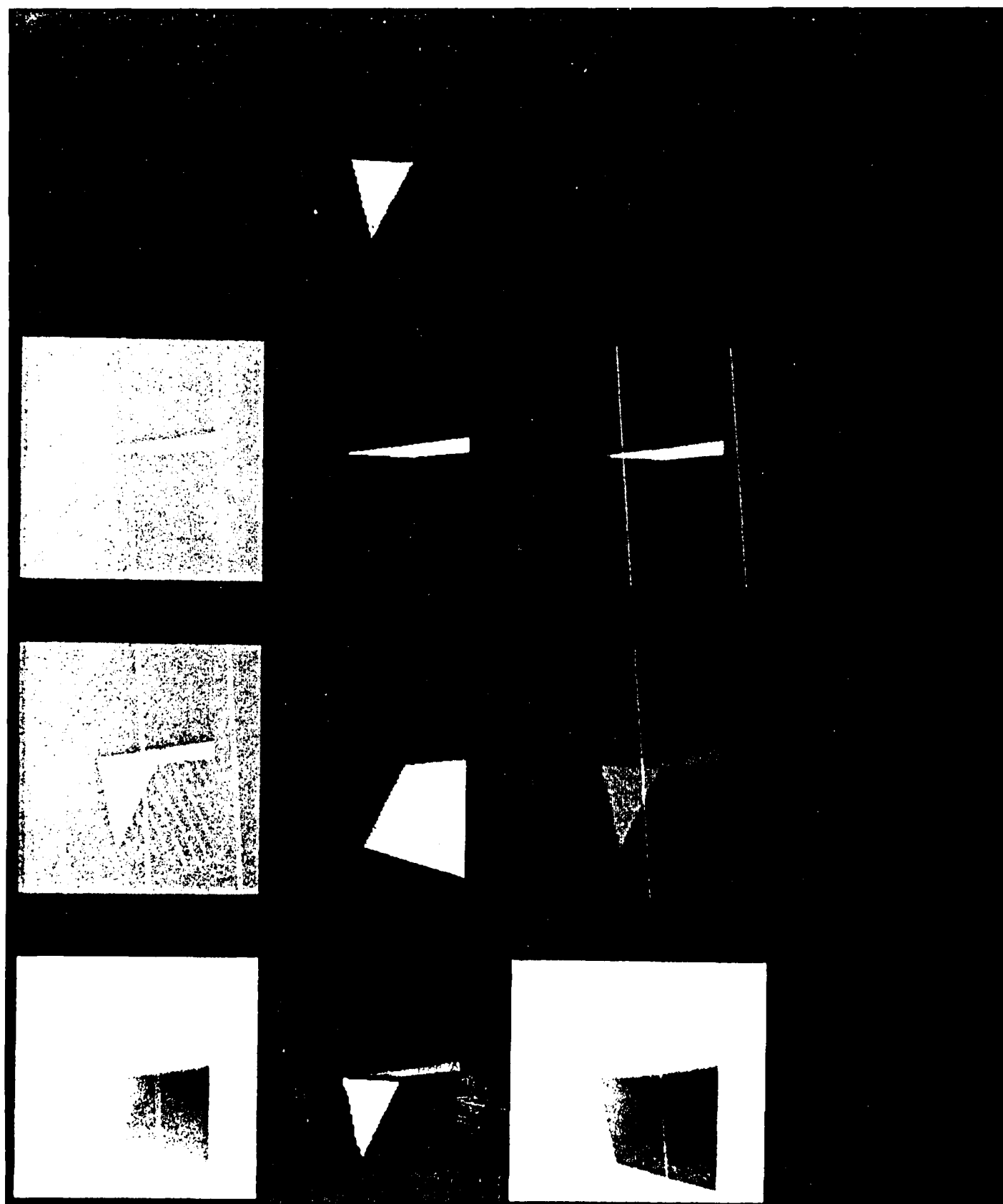


FIGURE 1

RESULT OF THE OPERATION FROM FIG. 1

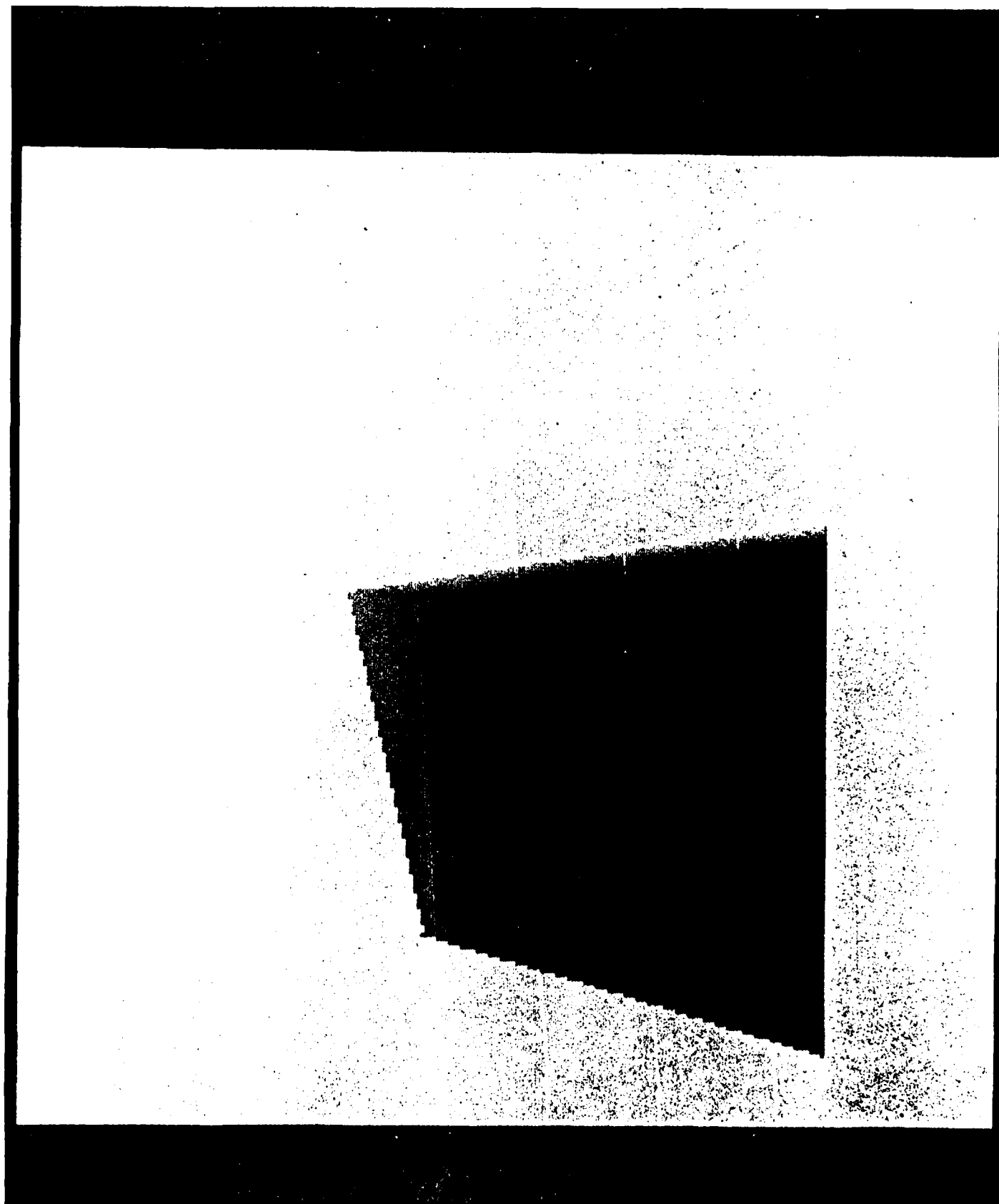


Plate 2

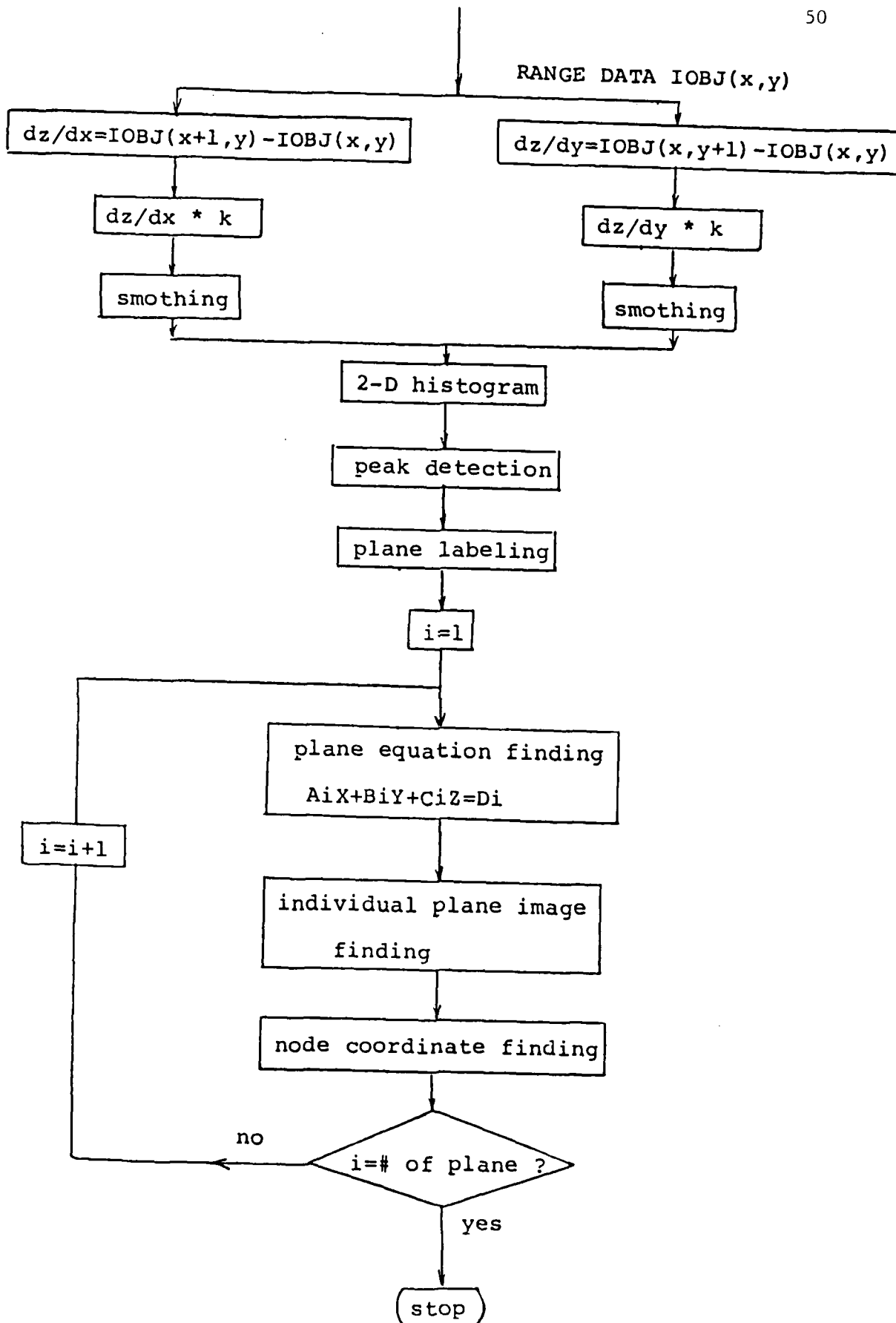
A simulated range image, mapped as gray levels. Darker pixels represent smaller distances

As an example, we chose a prism sitting on the X-Z plane as our reference object; it had six nodes and five planes. As a convex, plane-faced object, it may be represented by just the node coordinates, which will be used in later object matching procedures.

After a simple geometric transformation of the reference object the rotated reference object with known node coordinates and plane equations is chosen for range data generating. The X-Y plane is the sensor plane and the z values are the range values which would be represented in our range data image by the gray levels. Plate 2 shows the intensity range image displayed on the RAMTEK 9400. The darker the pixel on the image, the smaller the distance.

In order to examine the results of the range data recognition algorithm, we store the 3-D node coordinates and the parameters of the plane equations (i.e. A,B,C,D values in plane equation  $AX+BY+CZ=D$ , with D equal to either 1 or 0).

The artificial range data are then sent to the range data recognition system. The outputs of the system are the number of the planes (in this specific view), the plane equations and the node coordinates (visible in the view). One can compare the outputs with the original data which are used to generate the range data image to estimate the performances of the range data recognition system.



FLOWCHART OF PLANE-DESCRIPTION PROCEDURE

Fig 2

The object matching system receive the information from both the reference object and the single view. Comparing the node coordinates and plane equations, one can easily make the decision -- matching or mismatching.

### B.3 Range Data Recognition

The range data recognition algorithm starts with the range data image IOBJ(x,y), and contains the following three steps:

- (i). plane segmentation
- (ii). plane equation finding
- (iii) node coordinate finding

Fig. 2 is the flowchart of the range data recognition system.

We separate the planes in our single view image based on the different orientation of the individual plane. To find the orientation of the individual plane, it is necessary to compute the slopes in both X and Y directions (i.e. first partial derivative  $dz/dx$  and  $dz/dy$ ). Plates 3(a) and (b) show respectively the slopes  $dz/dx$  and  $dz/dy$ . The area which is brighter than the background has positive value of the first partial derivative, and the darker area has negative values. Since every plane has its own values of  $dz/dx$  and  $dz/dy$ , it is possible to separate the planes in the 2-D slope histogram (Fig. 3) by simple peak detection.



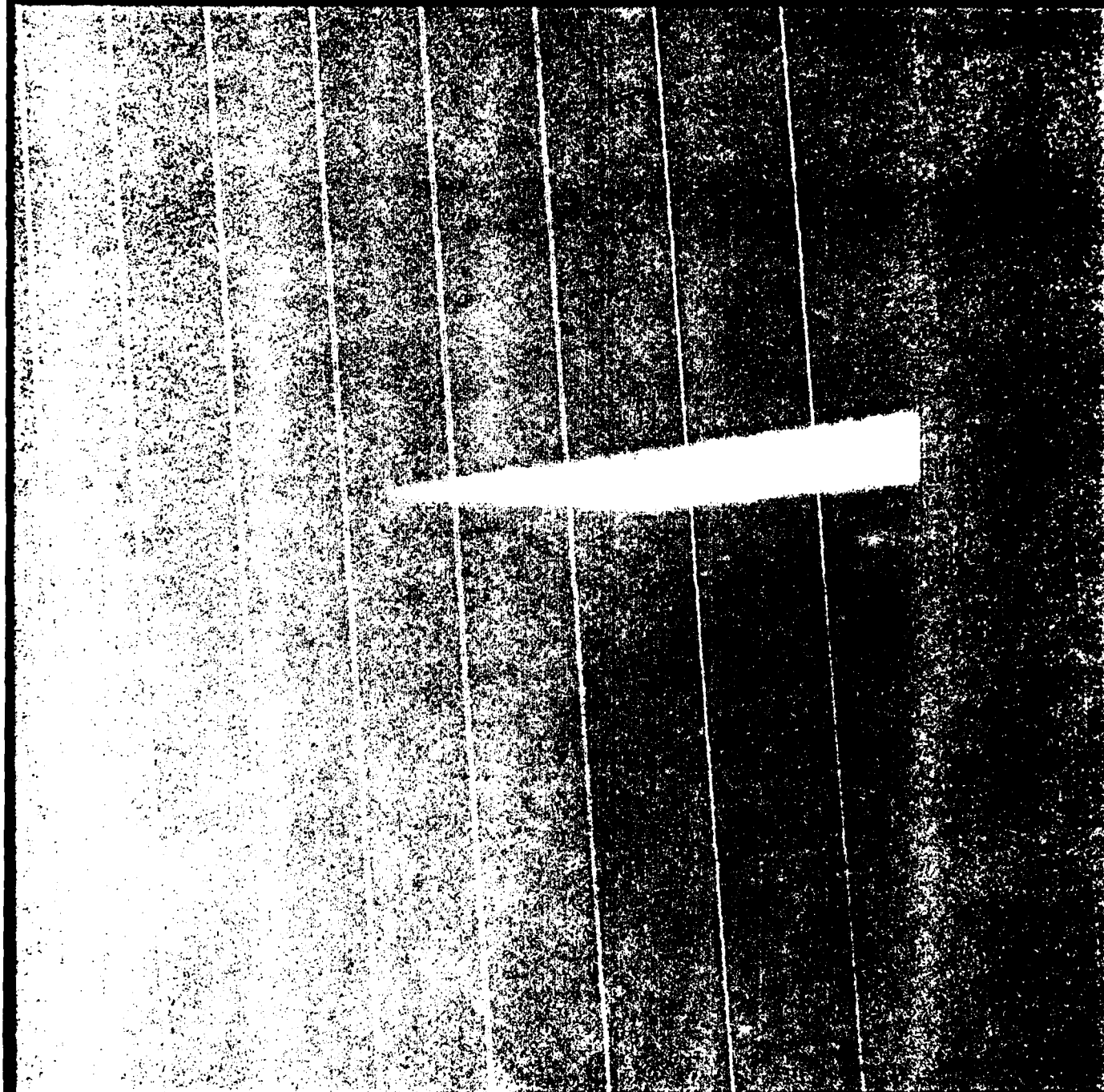
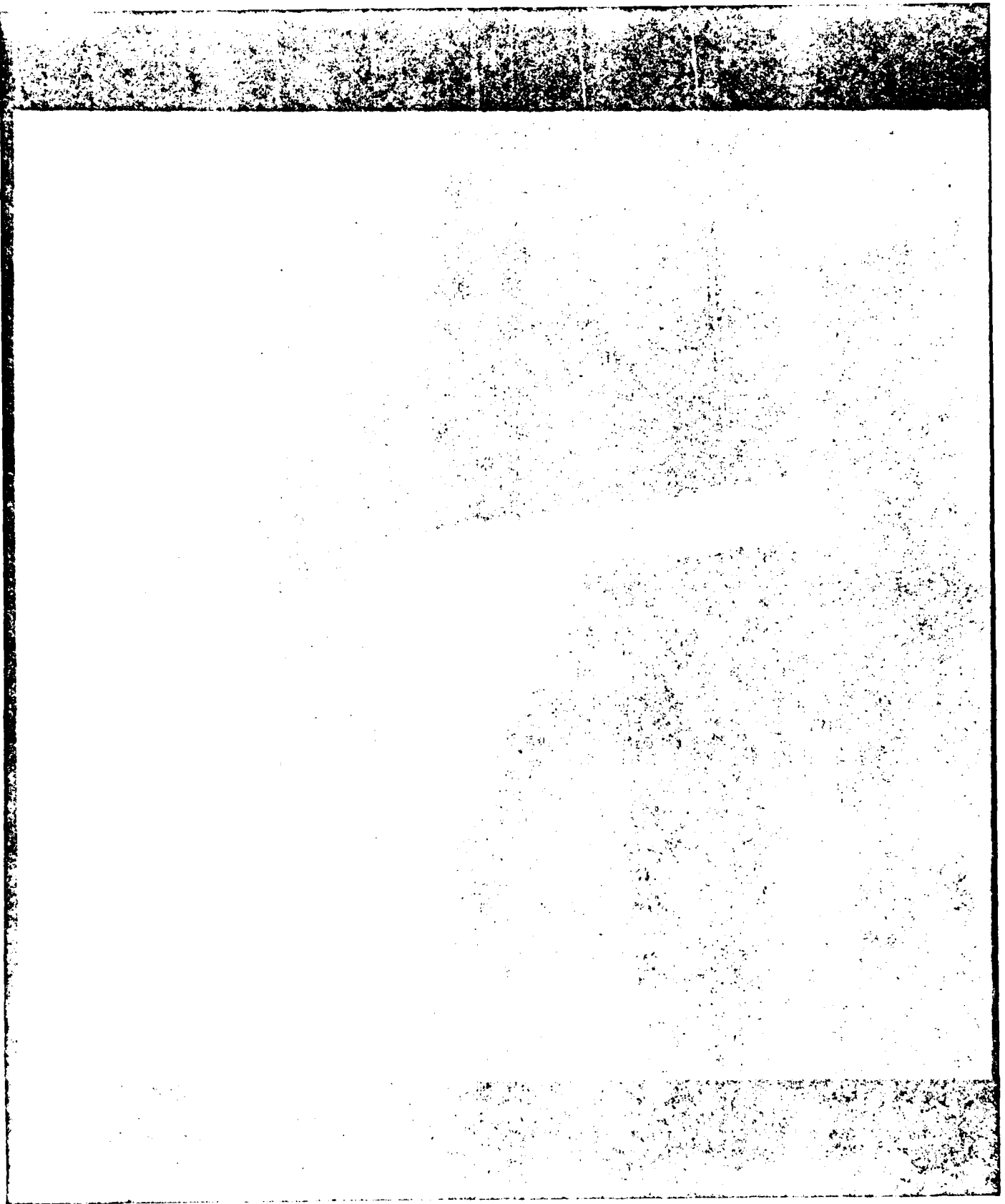


Plate 3

Images of slope mapped as intensity: (a)  $dz/dx$



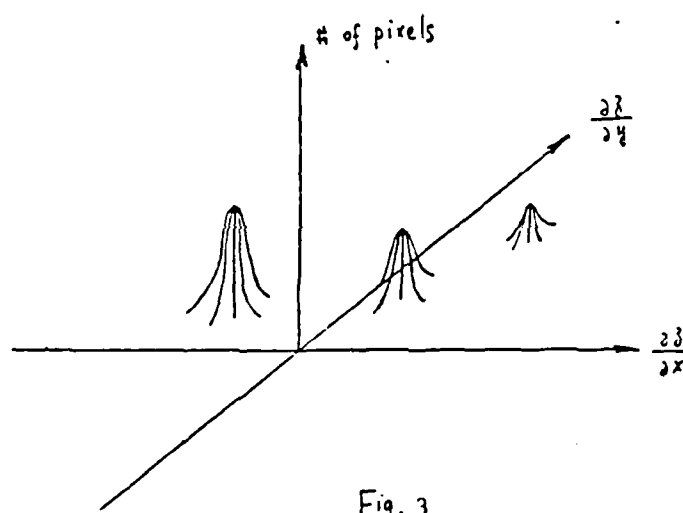


Fig. 3

X

SLOPE HISTOGRAM FOR IDENTIFICATION OF  
DISTINCT PLANES

An inherent problem in any digitized intensity image is the presence of discontinuities in the first derivative because of the limited gray levels. This problem causes the peak spreading and even overlap in our 2-D slope histogram. For instance, for the very shallow (nearly horizontal to X or Y axes) planes we might find slope value 0,1 in one plane and 0,-1 in another plane (see Fig. 4(a)) so that there would be an overlap in the slope histogram (Fig. 4(b)) and therefore no way to separate these two planes.

To solve the problem, we multiply the slope value by a factor  $k$  (say  $k=5$ ) to increase the number of the gray levels (Fig. 5(a)). After the smoothing procedure (Fig. 5(b)), one can easily find the separated peaks in the slope histogram (Fig. 5(c)). It is then no problem to separate these two planes.

Each strong peak in the 2-D slope histogram implies the presence of a distinct plane in our range data image. Taking the peak as the center and considering the 8 neighbors of the peak (see Fig. 6) we can label the pixels which have their  $dz/dx$  values between  $(dz/dx)_{\min}$  and  $(dz/dx)_{\max}$  and their  $dz/dy$  values between  $(dz/dy)_{\min}$  and  $(dz/dy)_{\max}$  as having the same value (see Plate 4).

Because of the slope smoothing procedure the boundaries of the planes were blurred. The pixels around the boundaries were reasonably rejected during the plane labeling. The planes appearing in Plate 4 are smaller than those they represent, but this margin makes for more-confident estimates of the planes' equations.

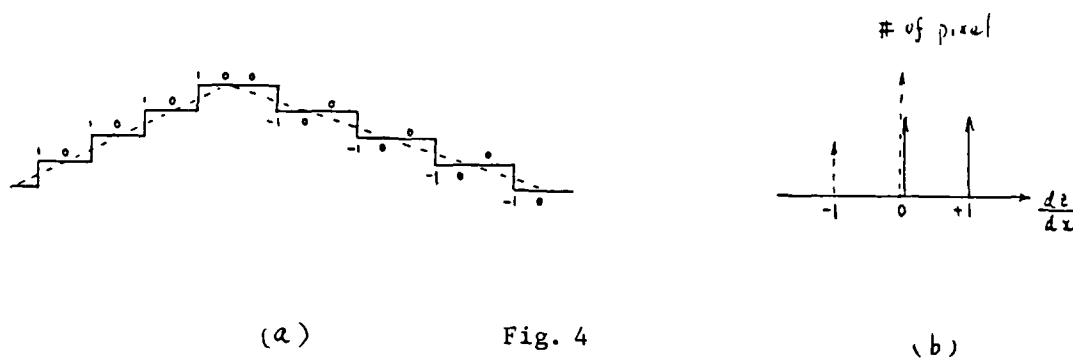


Fig. 4

EFFECT OF LIMITED DYNAMIC RANGE. (a) Cross-Section of two planes meeting at shallow angle (b) The corresponding histogram, and its poorly-resolved peaks

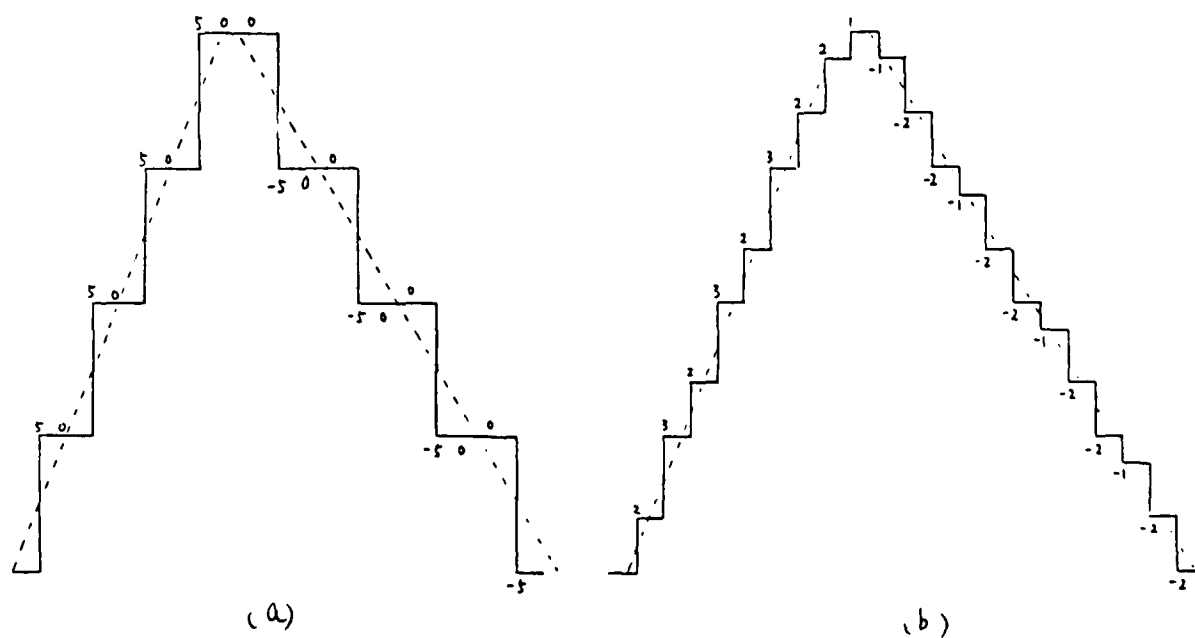
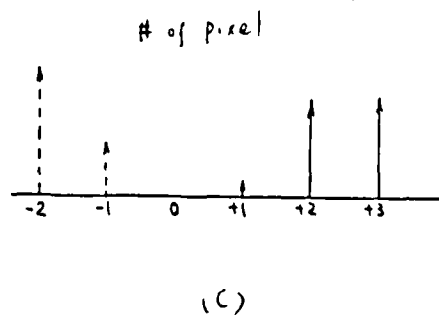


Fig. 5

ENHANCEMENT OF THE EXAMPLE OF FIG. 4

- (a) multiplication by a constant  
(b) smoothing (c) improved histogram



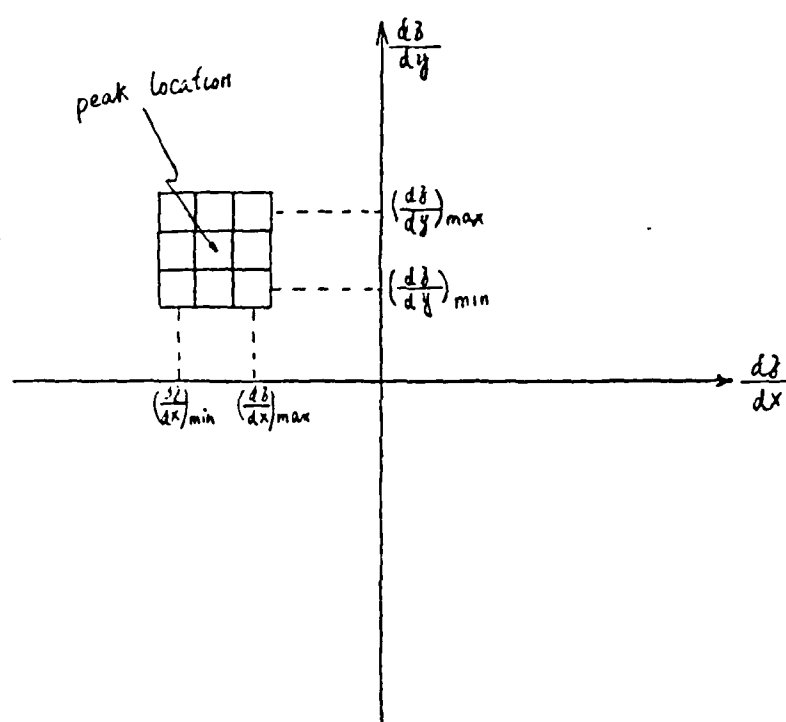


Fig. 6

NEIGHBORHOODS OF A PEAK IN THE 2-D SLOPE HISTOGRAM DEFINE  
MAXIMA AND MINIMA

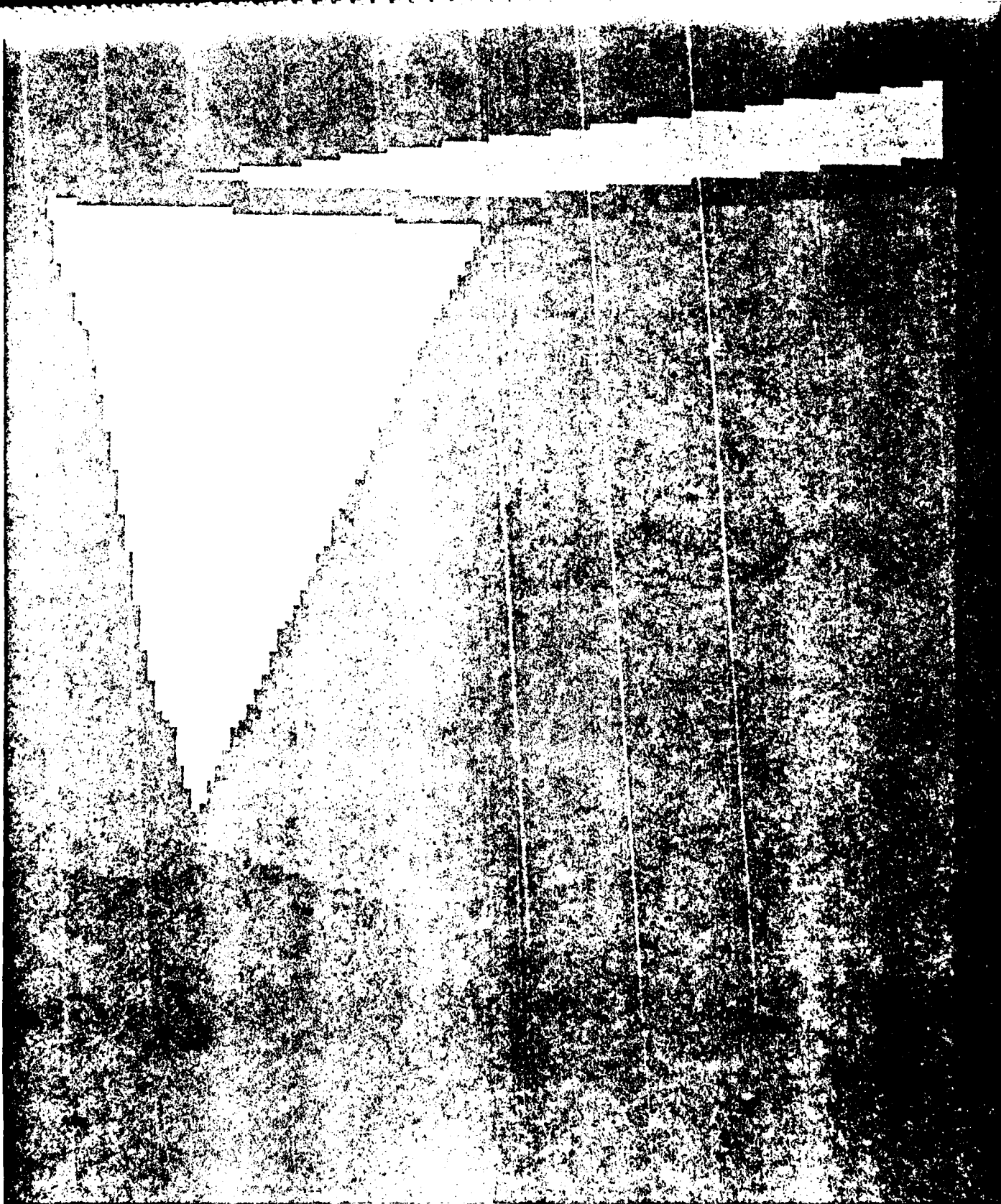


Plate 4 Identification in the range image of the distinct  
plane: pixels with slopes within the ranges defined by  
Figure 6 are shown in a gray level.

Picking up 3 points on a labeled plane (choosing them to be as far apart as possible), one can find the plane equation in the form of  $AX+BY+CZ=D$  ( $D=1$  or  $0$ ).

Once we learn the plane equation, we can readily solve several problems:

(i) to find the relationship between the planes by their plane equation parameters  $A_i, B_i, C_i, D_i$ .

(ii) to find the individual plane images (i.e. the pixels whose  $IOBJ(x,y)$  value satisfies the plane equation). Plates 5(a),(b),(c) show the individual planes which were finally found. They are exactly the original planes.

(iii) searching the boundary pixels in the individual plane image and using the chain code as the indicator of the directions, one can find the turning points in the boundaries (i.e. the nodes of the object) in terms of the changes in the chain code.

The outputs of our range image recognition program -- the number of planes, the plane equations, and the visible-node coordinates -- offer enough information for the object matching procedure.

#### B.4 Object Matching

The object matching program will answer the question --- is this single view a specific view of our reference object? The matching procedure includes the following three steps (see Fig. 7):

(i) bottom-node matching



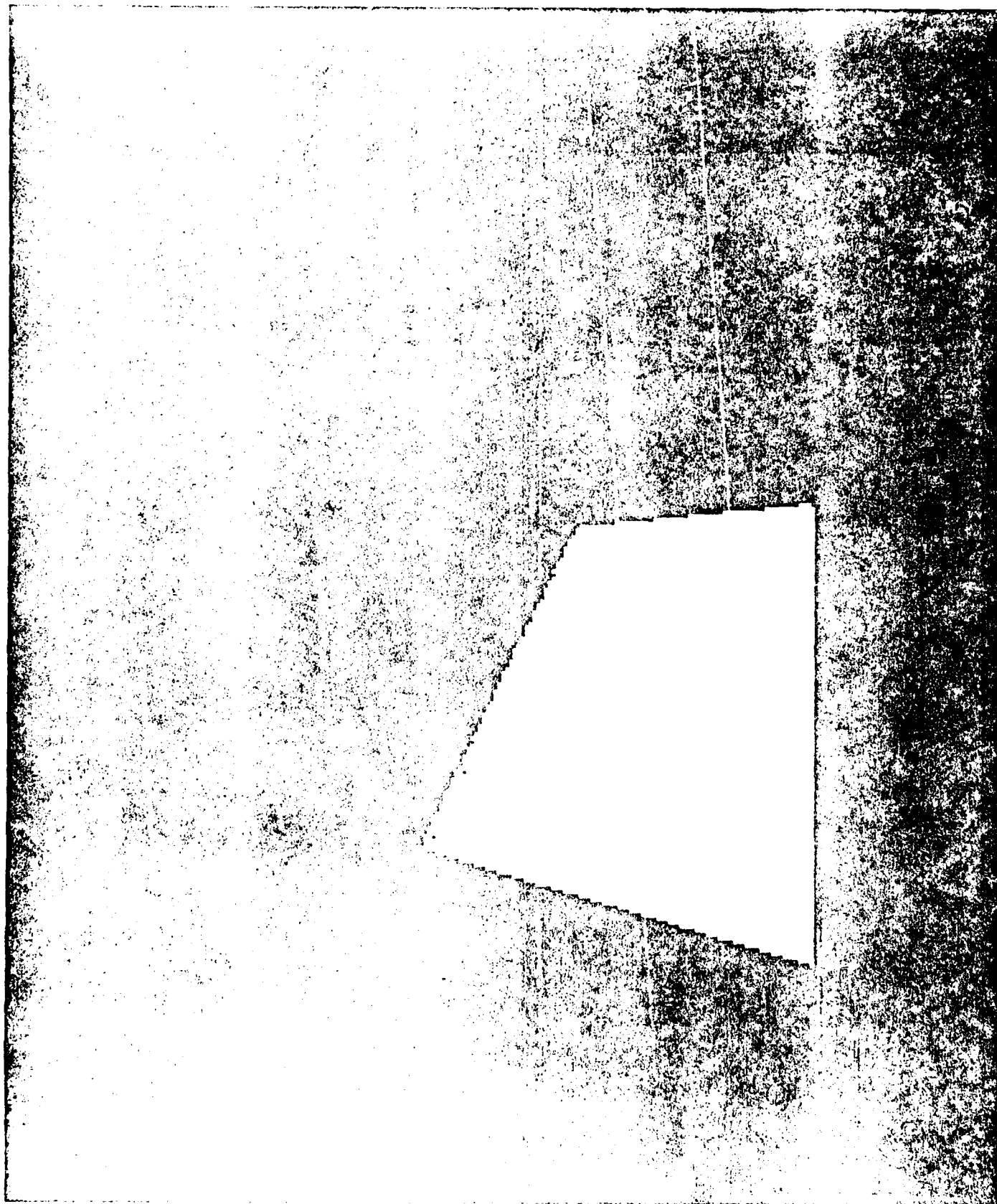
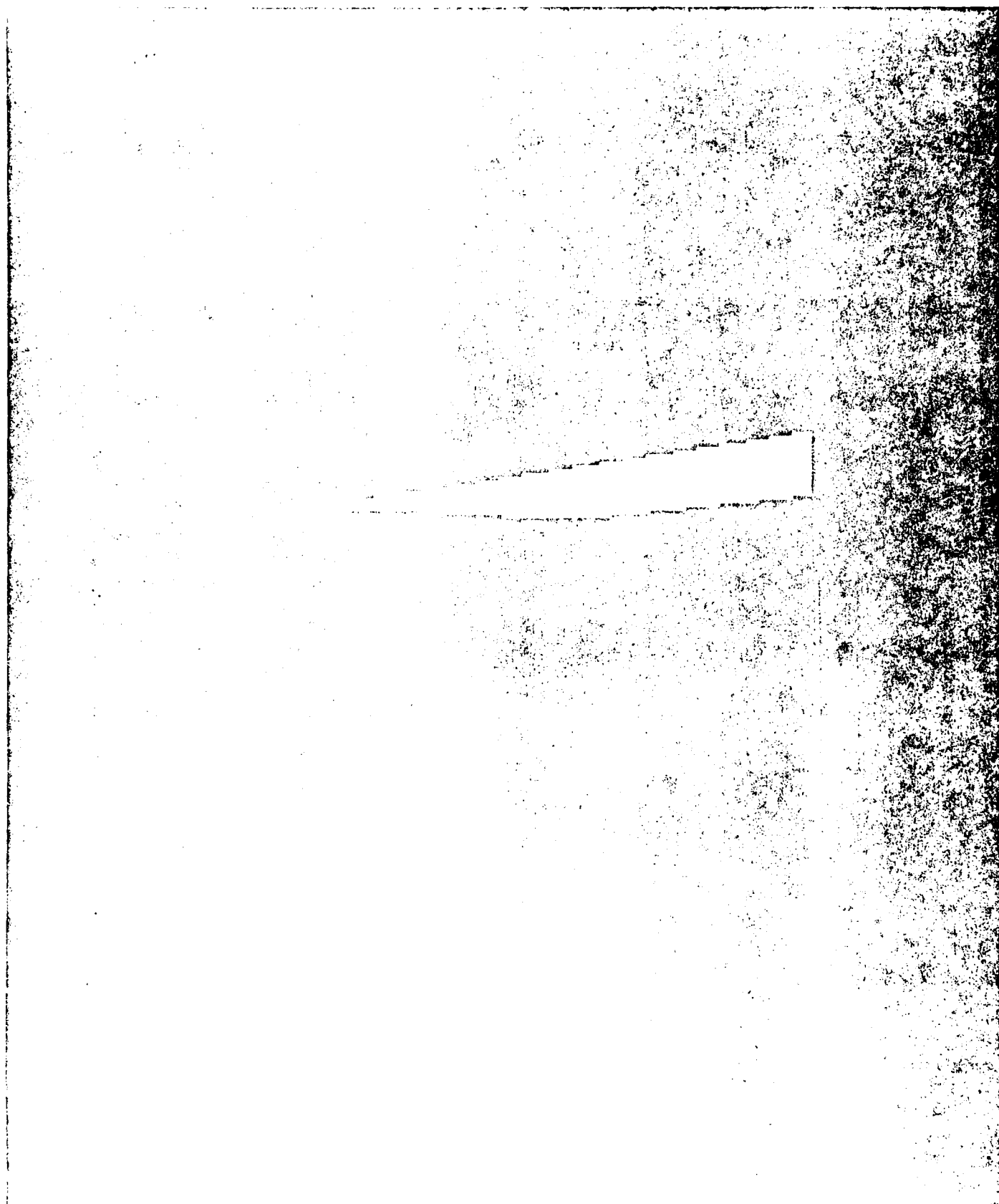
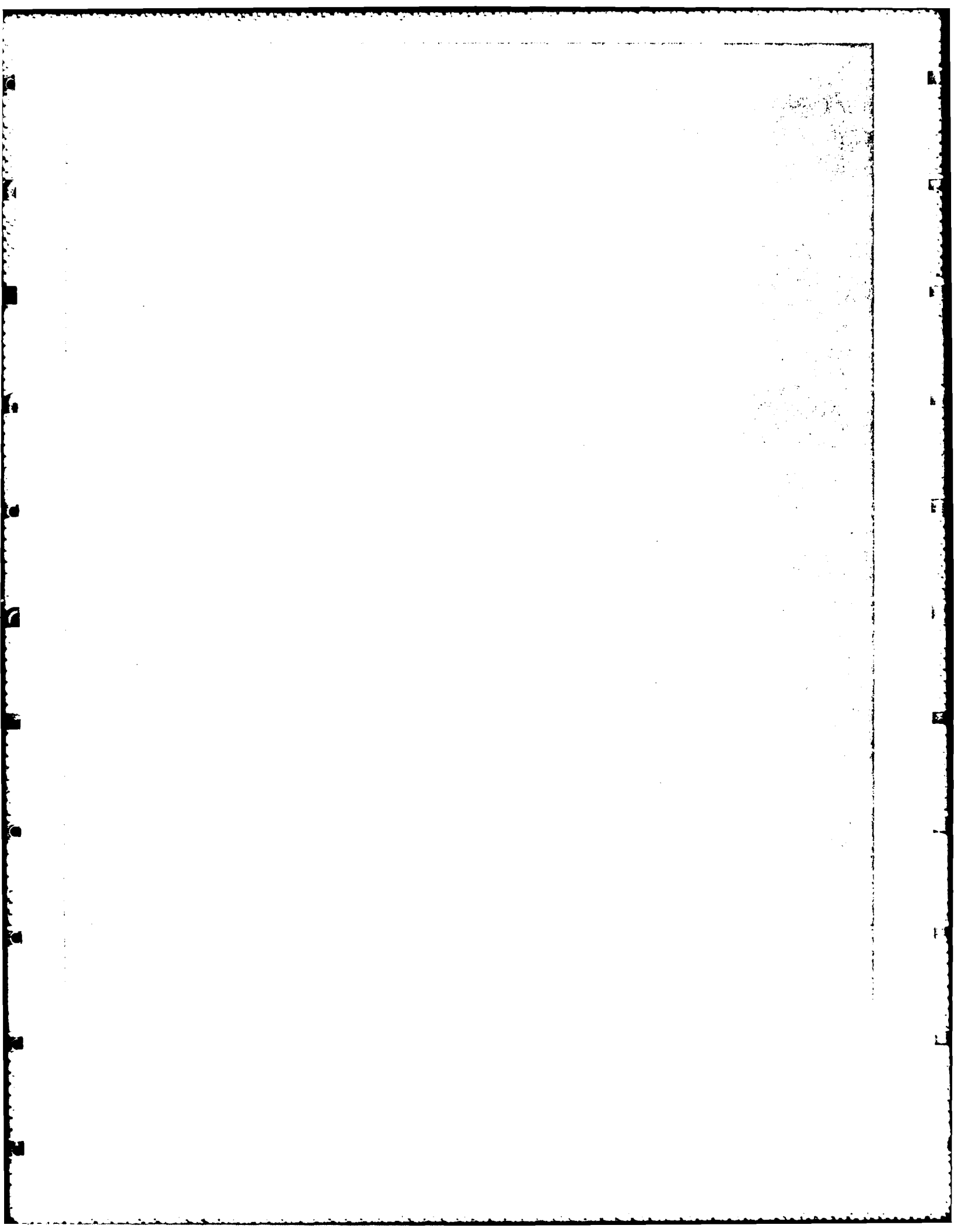


Figure 1. Three views of the specimen: (a), (b), and (c). They are the same specimen as in Figure 2.





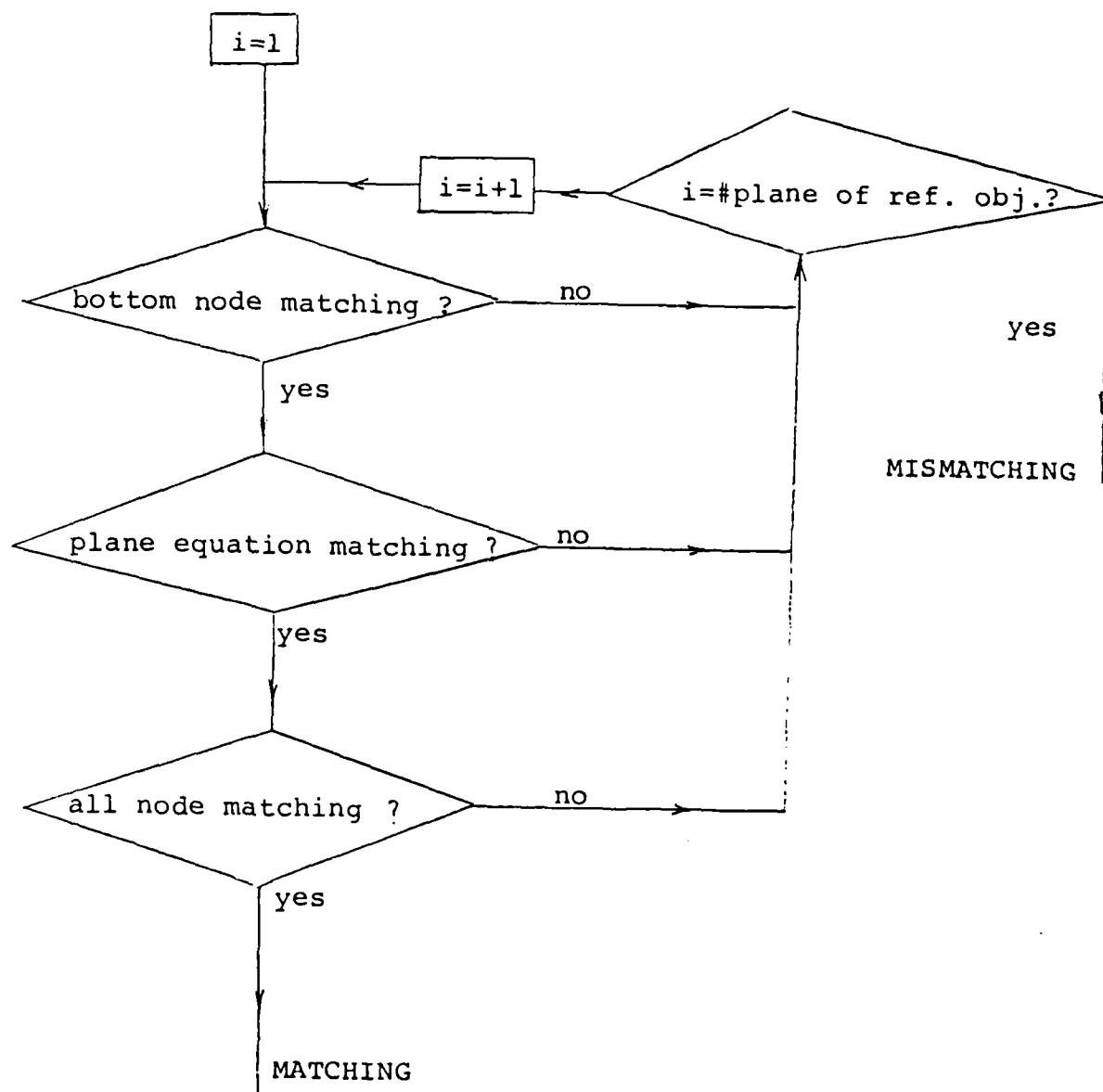


Fig. 7

FLOWCHART OF THE OBJECT-MATCHING PROCEDURE

(ii) plane-equation matching

(iii) all-node matching

The node coordinates and plane equations of the single view are found from the range image recognition program, and those of the reference object are assumed known a priori.

It was assumed that the object we are observing has its bottom plane parallel to the X-Z plane. Therefore, one can choose one of the planes of our reference object as the bottom plane and let it be parallel to X-Z plane (one of the possible positions) when we start the matching procedure.

A parallel transformation is needed for both reference object and the object being observed in order to compare their node coordinates and plane equations. Putting a bottom node at the origin (see Fig. 8) we can first compare the coordinates of the bottom node. It means to compute the total error in 3-D coordinates

$$e_{b.n} = \sum_{i=1}^N \left[ (x_{ri} - x_{oi})^2 + (y_{ri} - y_{oi})^2 + (z_{ri} - z_{oi})^2 \right]^{1/2}$$

where N is the number of nodes:

$x_{ri}, y_{ri}, z_{ri}$  are the node coordinates of the reference object;

$x_{oi}, y_{oi}, z_{oi}$  are the node coordinates of the observed object.

Usually a rotation of the reference object about the Y axis has to be done before we learn whether the bottom nodes are matching or not. If it is not, we would put another bottom node of our reference object (but in the same position) to the origin and

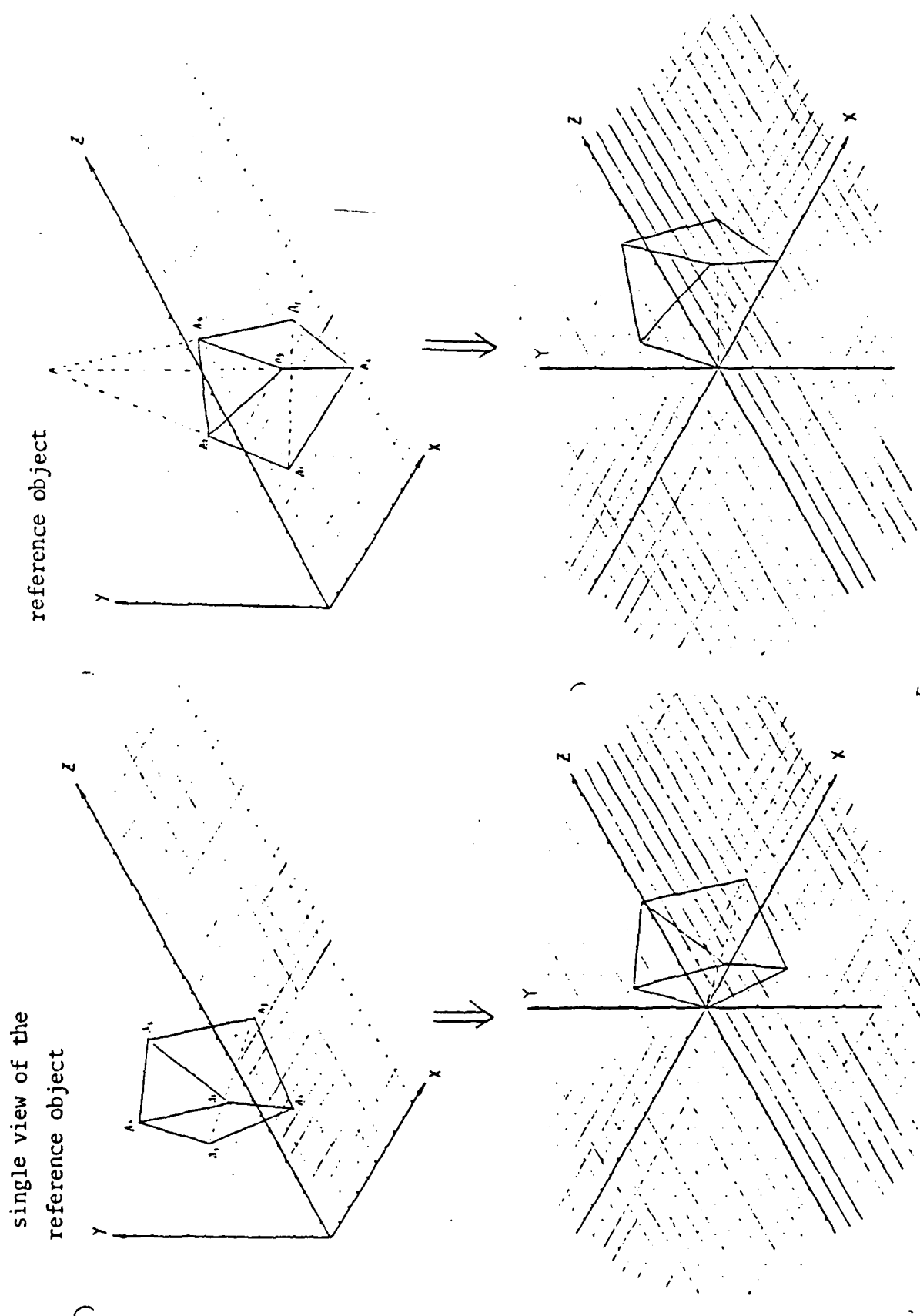


Fig. 8

EXAMPLE OF THE OBJECT-MATCHING PROCEDURE

compare it to the observed object again. If it is still not matching after we try all bottom nodes in this position at the origin, we will change the position of the reference object and go through the same bottom node matching procedure again. If it does match we will proceed with the plane equation matching program. One can simply compute the difference between the parameters of the plane equations

$$e_{p.e.} = \sum_{i=1}^M \left[ (A_{ri} - A_{oi})^2 + (B_{ri} - B_{oi})^2 + (C_{ri} - C_{oi})^2 + (D_{ri} - D_{oi})^2 \right]^{1/2}$$

where  $m$  is the number of planes, and

$A_{ri}, B_{ri}, C_{ri}, D_{ri}$  are the parameters of the plane equation of reference object;

$A_{oi}, B_{oi}, C_{oi}, D_{oi}$  are the parameters of the plane equations of observed object.

Finally, we invoke the all-node matching program if the bottom node matching and the plane equation matching have passed. The goal of the all-node matching program is to check:

(i) if all nodes we found from the range image can find their partners in the reference object;

(ii) if all extra nodes in the reference object are behind the surface of our specific view.

## B.5 Results and Conclusions

To examine the simulation system, we compared the original node coordinates and plane equations which we used to generate the range image and those we finally found using the range image recognition system. The results are presented in Table I and Table II respectively. The total error for all 18 node coordinates was 10.87 and the error between the coordinates of the plane equations was 0.01268. This results show the good performance of our recognition system. In addition, the matching procedure has an advantage of requiring both rather small memory space and the amount of the computation because the whole matching procedure does not require image-sized data operations. (Appendix E contains listings of the software used.)

## IV. Conclusions and implications for the Navy

The results presented here indicate that structured light is a feasible technique for real-time ranging. Equally, regardless of how a range image is acquired, useful descriptions of objects can be extracted from such an image. The DAD box appears to be a fast processor that can use look-up tables to compute range in real-time, and should be considered for integration into structured-light systems.

Clearly, further work must be done to examine the mechanical aspects of sweeping the masks past the light source, and the intensity of the light must be augmented. (An alternative to increasing the intensity of the source is the use of a photomultiplier at the camera



Table I - RESULTS OF PLANE-EQUATION DETERMINATION

| # of plane | plane equations                       |                                       |
|------------|---------------------------------------|---------------------------------------|
|            | to generate the data                  | we finally found                      |
| plane #1   | $0.00556X - 0.00148Y + 0.00963Z = 1$  | $0.00553X - 0.00150Y + 0.00968Z = 1$  |
| plane #2   | $-0.00050X + 0.01395Y - 0.01087Z = 1$ | $-0.00053X + 0.01387Y - 0.01070Z = 1$ |
| plane #3   | $0.00695X + 0.00096Y - 0.00100Z = 1$  | $0.00696X + 0.00095Y - 0.00101Z = 1$  |

Table II - RESULTS OF NODE-COORDINATE DETERMINATION

| # of node | node coordinates     |                  |
|-----------|----------------------|------------------|
|           | to generate the data | we finally found |
| A1        | 20, 50, 100          | 20, 50, 100      |
| A2        | 141, 50, 30          | 140, 51, 31      |
| A3        | 156, 50, 127         | 156, 50, 132     |
| A4        | 50, 150, 98          | 51, 150, 98      |
| A5        | 135, 110, 43         | 134, 111, 44     |
| A6        | 138, 170, 119        | 137, 169, 119    |

input. Efforts were made in this direction using an NBS-supplied low-light-level [LLL] camera. Irreconcilable optical incompatibilities between the LLL camera and the CID camera, however, made it impossible to make a realistic test of the benefits of such an approach.)

Possible disadvantages of this kind of system include the need for a very fine slit to reduce the size of the region of uncertainty cast by the masks. It is the requirement for the fine slit that leads to the need for a high-intensity light source (as noted above). The system, as it uses visible light, must operate in a darkened environment and hence is probably not suitable where human activity is also involved.

In many manufacturing-related applications, however, structured light is likely to be very useful. Opportunities include inspection tasks: for completeness of structure, or quality of surface finish; and assembly tasks: for parts matching and orientation. Many of the precision and/or hazardous manufacturing and handling responsibilities of the Navy appear to be reasonable candidates for adoption of this technology, which requires of necessity a relatively static platform for installation of the equipment. It is thus far better suited to the manufacturing and quality control environments than, for example, to a mobile-robot application.

Several research papers which bear on the range of prospective applications of this kind of a system appear in Appendix F.

## APPENDIX A

## CALCULATION AND DISPLAY OF VISIBLE REGION

Camera focal point assumed to be aligned with the first row of the CID plane. (Physical arrangement was alignment with center of CID plane; difference in visible region is small.)

01181

```
480 INPUT "WHAT IS THE DESIRED CAMERA TILT?";C
492 PRINT ""
484 PRINT ""
486 PRINT ""
488 PRINT "D","RO","YO","KA","YA","RB","RC"
490 C1 = (C / 180) * (3.141592654)
500 A = .042
510 F1 = .142
520 B = .008784
530 F2 = .025
540 C2 = 61N (B / F2)
545 FOR D = .5 TO 2.5 STEP .5
550 RO = D / ((A / F1) + (1 / TAN (C1 - C2)))
560 YO = D / (1 + ((F1 / A) / (TAN (C1 - C2))))
570 KA = D / ((A / F1) + (COS (C1) / SIN (C1)))
580 YA = D / (1 + ((F1 * COS (C1)) / (A * SIN (C1))))
590 RB = D * (TAN (C1 - C2))
600 IF C = 90 THEN 630
610 RC = D * (TAN (C1))
620 PRINT D,RO,YO,KA,YA,RB,RC
625 GOTO 650
630 RS$ = "INFINITY"
640 PRINT D,RO,YO,KA,YA,RB,RC$
650 NEXT D
660 END
```

ORUN

WHAT IS THE DESIRED CAMERA TILT?

| D   | KU         | YU         | KA         | YA  | KB         | RC       |
|-----|------------|------------|------------|-----|------------|----------|
| 0   | .772636733 | .228526358 | 1.69047619 | .5  | 1.42304189 | INFINITY |
| 1   | 1.54527347 | .457052715 | 3.38095238 | 1   | 2.84608379 | INFINITY |
| 1.5 | 2.3179102  | .685579073 | 5.07142858 | 1.5 | 4.26912569 | INFINITY |
| 2   | 3.09054693 | .91410543  | 6.76190476 | 2   | 5.69216758 | INFINITY |
| 2.5 | 3.86318367 | 1.14263179 | 8.45238095 | 2.5 | 7.11520948 | INFINITY |

BRUN

WHAT IS THE DESIRED CAMERA 1111775

| D   | RU          | YU         | KA         | YA         | RB         | RC        |
|-----|-------------|------------|------------|------------|------------|-----------|
| 0.5 | .310490829  | .150990245 | .886959118 | .262340021 | .731341777 | 1.8660254 |
| 1   | 1.020998166 | .30198049  | 1.77391824 | .524680041 | 1.46268356 | 3.7320508 |
| 1.5 | 1.53147249  | .452970735 | 2.66087735 | .787020062 | 2.19402533 | 5.5980762 |
| 2   | 2.04196332  | .60396098  | 3.54783647 | 1.04936008 | 2.92536711 | 7.4641016 |
| 2.5 | 2.55245414  | .754931226 | 4.43479559 | 1.3117001  | 3.65670889 | 9.330127  |

TRUN

WHAT IS THE DESIRED CAMERA TILT?

| D   | RO         | YU         | RA         | YA         | RB         | RC         |
|-----|------------|------------|------------|------------|------------|------------|
| 0   | .342272303 | .10123553  | .572655746 | .169377051 | .429166251 | .866025403 |
| 1   | .684545011 | .20247106  | 1.14531149 | .338754103 | .858332502 | 1.73205081 |
| 1.5 | 1.02681752 | .303706589 | 1.71796724 | .508131154 | 1.28749875 | 2.59807621 |
| 2   | 1.36909002 | .404942119 | 2.29062298 | .677508206 | 1.716665   | 3.46410161 |
| 2.5 | 1.71136253 | .506177649 | 2.86327873 | .846885257 | 2.14583126 | 4.33012702 |

JKUN

WHAT IS THE DESIRED CAMERA TILT?

| D   | KU         | YU          | KA         | YA         | RB         | RC  |
|-----|------------|-------------|------------|------------|------------|-----|
| .5  | .21015917  | .0621597547 | .685869565 | .114130435 | .239995264 | .5  |
| 1   | .420318341 | .124319509  | .771739131 | .22826087  | .479990528 | 1   |
| 1.5 | .630677512 | .186479264  | 1.1576087  | .342391304 | .719985792 | 1.5 |
| 2   | .840636682 | .248639019  | 1.54347826 | .456521739 | .959981056 | 2   |
| 2.5 | 1.05079585 | .310798773  | 1.92934783 | .570652174 | 1.19997632 | 2.5 |

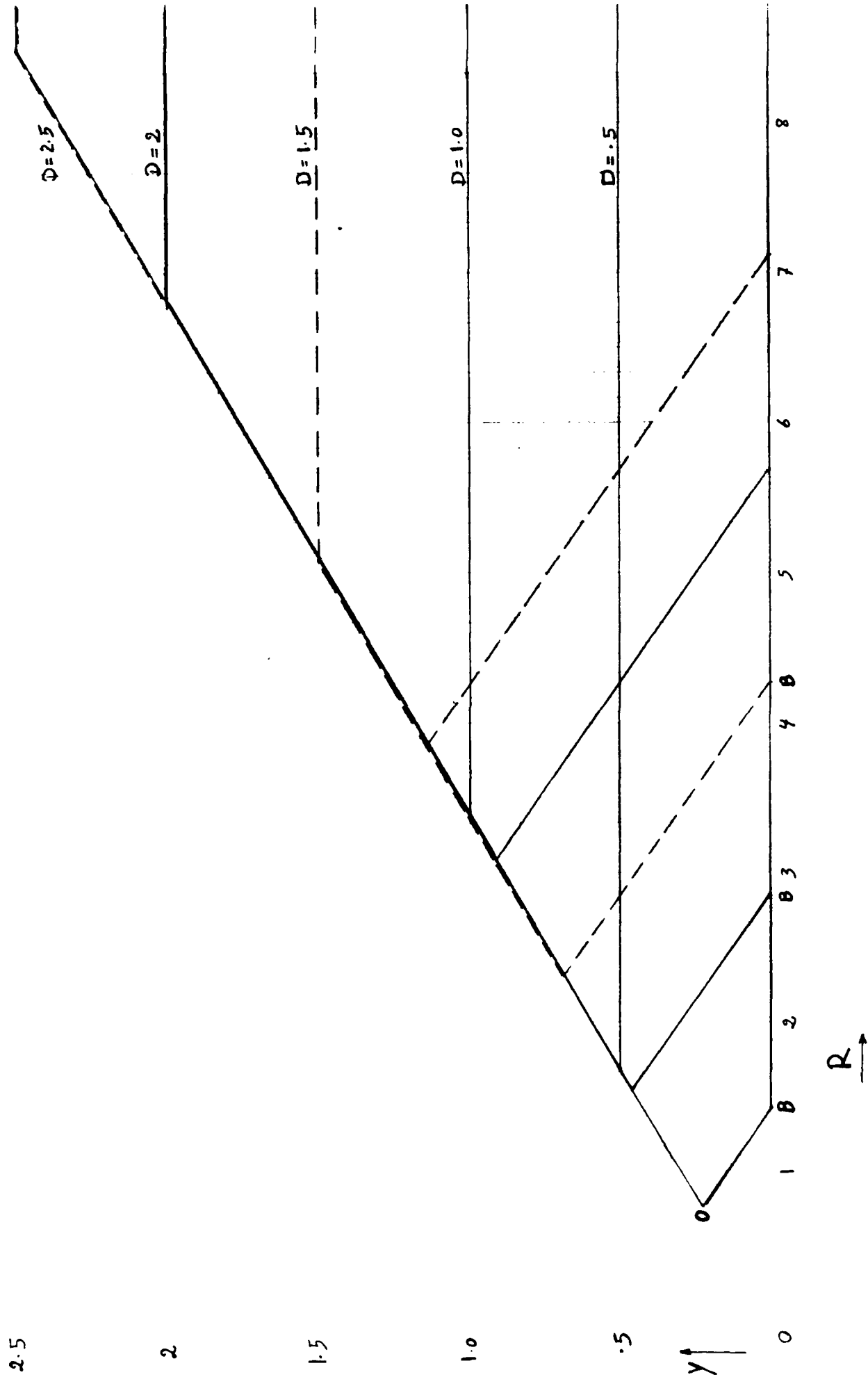


DEON

What is the desired camera tilt?

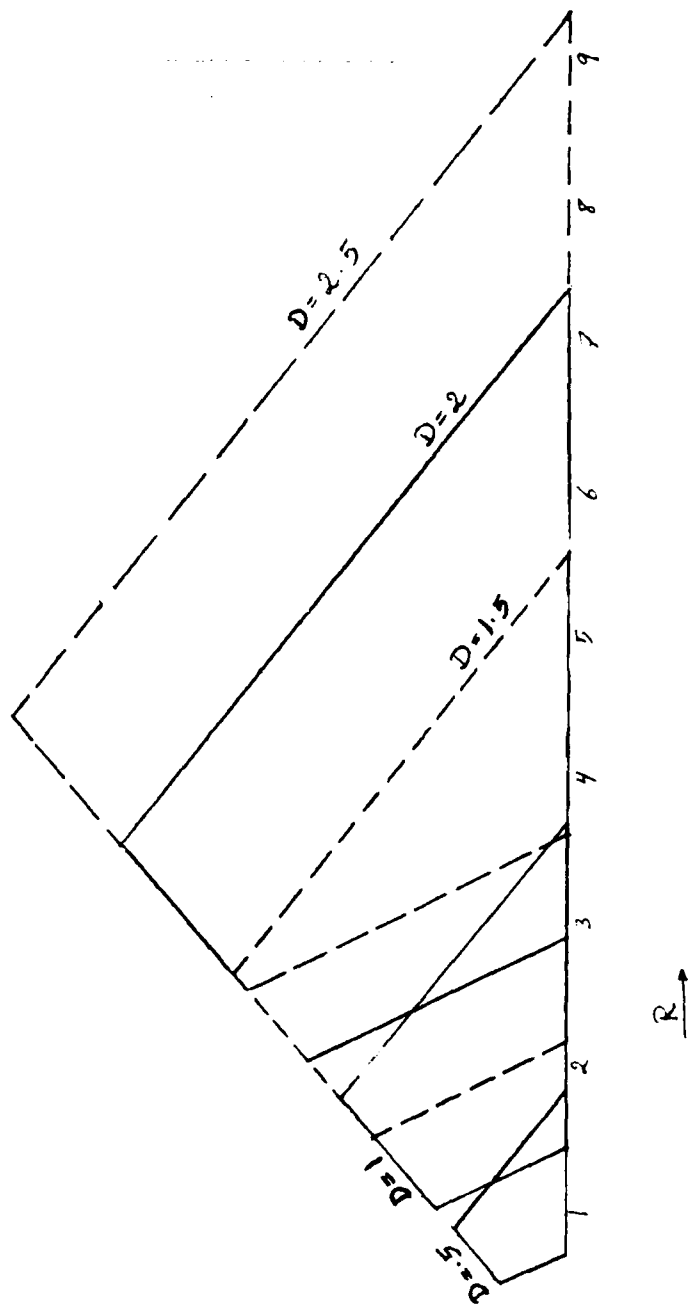
| D   | RO          | YO          | KA         | YA          | RB          | RC         |
|-----|-------------|-------------|------------|-------------|-------------|------------|
| 1   | .0889935/01 | .0263220419 | .246569545 | .0729290203 | .0939388973 | .288675135 |
| 1   | .17798714   | .0526440837 | .49313909  | .145858041  | .187877795  | .577350269 |
| 1.0 | .26698071   | .0789661256 | .739708635 | .218787061  | .281816692  | .866025404 |
| 2   | .35597428   | .105288167  | .98627818  | .291716081  | .375755589  | 1.15470054 |
| 2.0 | .444967851  | .131610209  | 1.23284773 | .364645102  | .469694486  | 1.44337567 |

VISIBLE REGION FOR CAMERA TILT OF  $90^\circ$   
FOR DIFFERENT PROJECTOR CAMERA DISTANCES



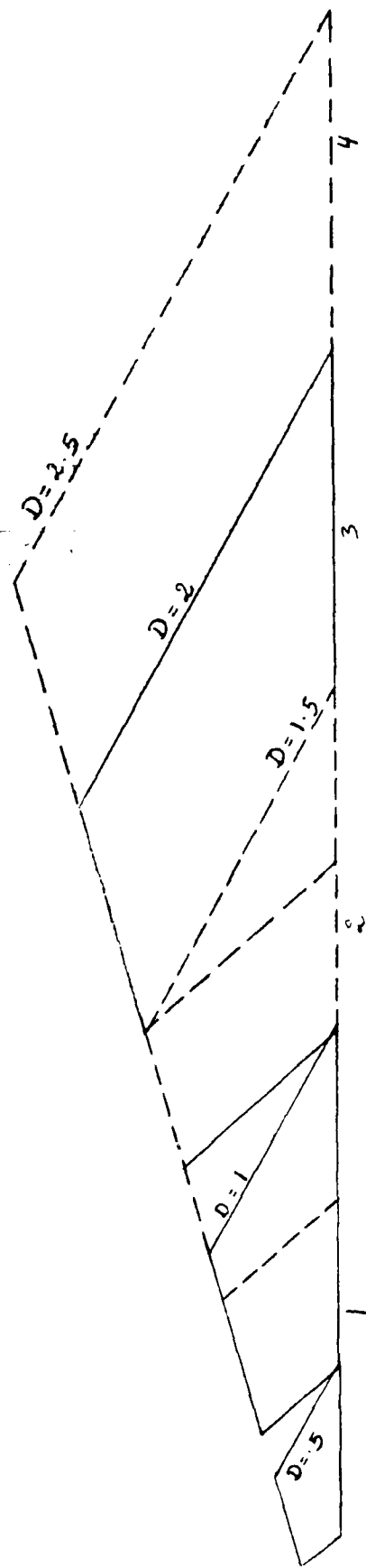
CAMERA VIEW = 75°

$$SCALE = \frac{Y}{R} = 3$$



CAMERA TILT =  $60^\circ$

$$\text{SCALE} = \frac{Y}{R} = 1$$



CAMERA TILT =  $45^\circ$

2.5

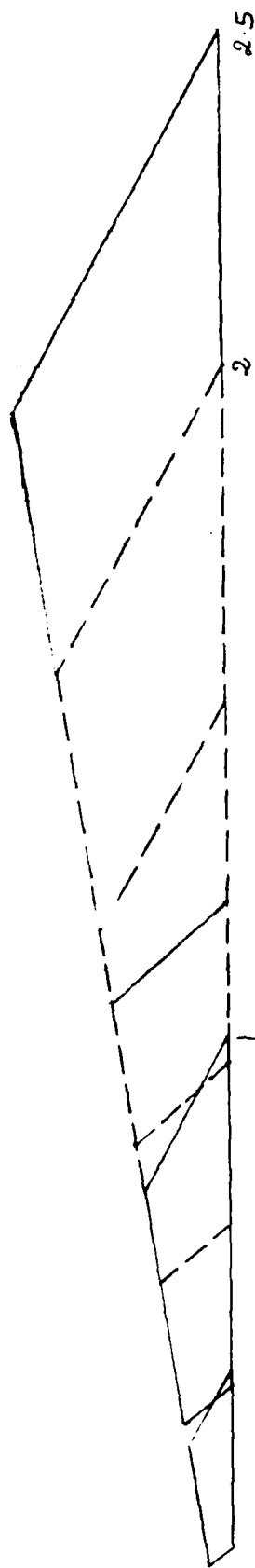
2

1.5

1

5

0



## APPENDIX B

## CALCULATION AND PLOTS OF THEORETICAL PRECISION OF RANGE ESTIMATES

A.

The following plot shows the sensitivity of the computed range by the system at a specific camera tilt (c), object range (R), camera-projector (d), and the change in range ( $\Delta R$ ) versus the change in object vertical location:

a)  $\Delta R = .01$   
d=1  
R=1 } solid black line  
c=90° }

d=1  
R=1 } Red line  
C=75° }

d=2  
R=1 } dashed line  
c=90° }

b)  $\Delta R = .001$   
same as above

c)  $\Delta R = .0001$   
same as above

A.

$R'$   
Meters

$5 \times 10^{-3}$

(a)  
 $\Delta R = .01$

$10^{-4}$

(b)  
 $\Delta R = .001$

$d=1 \quad R=1 \quad 90$

$10^{-5}$

(c)  
 $\Delta R = .0001$

75

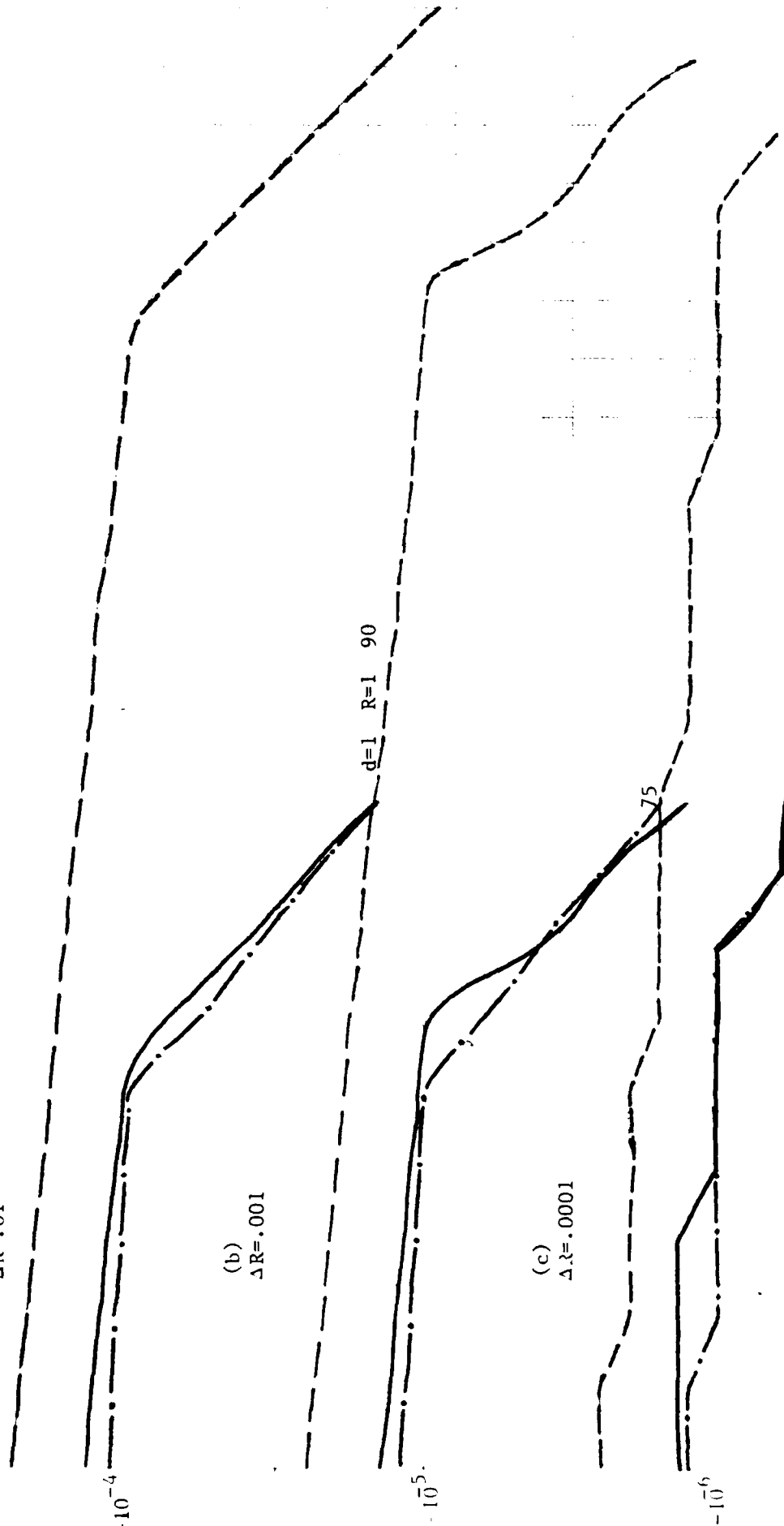
$10^{-6}$

$5 \times 10^{-7}$

OBJECT HEIGHT ABOVE PLANE OF SLIT, Y, METERS

0.1 0.2 0.3 0.4 0.5

1.4 1.5 1.6 1.7 1.8 1.9



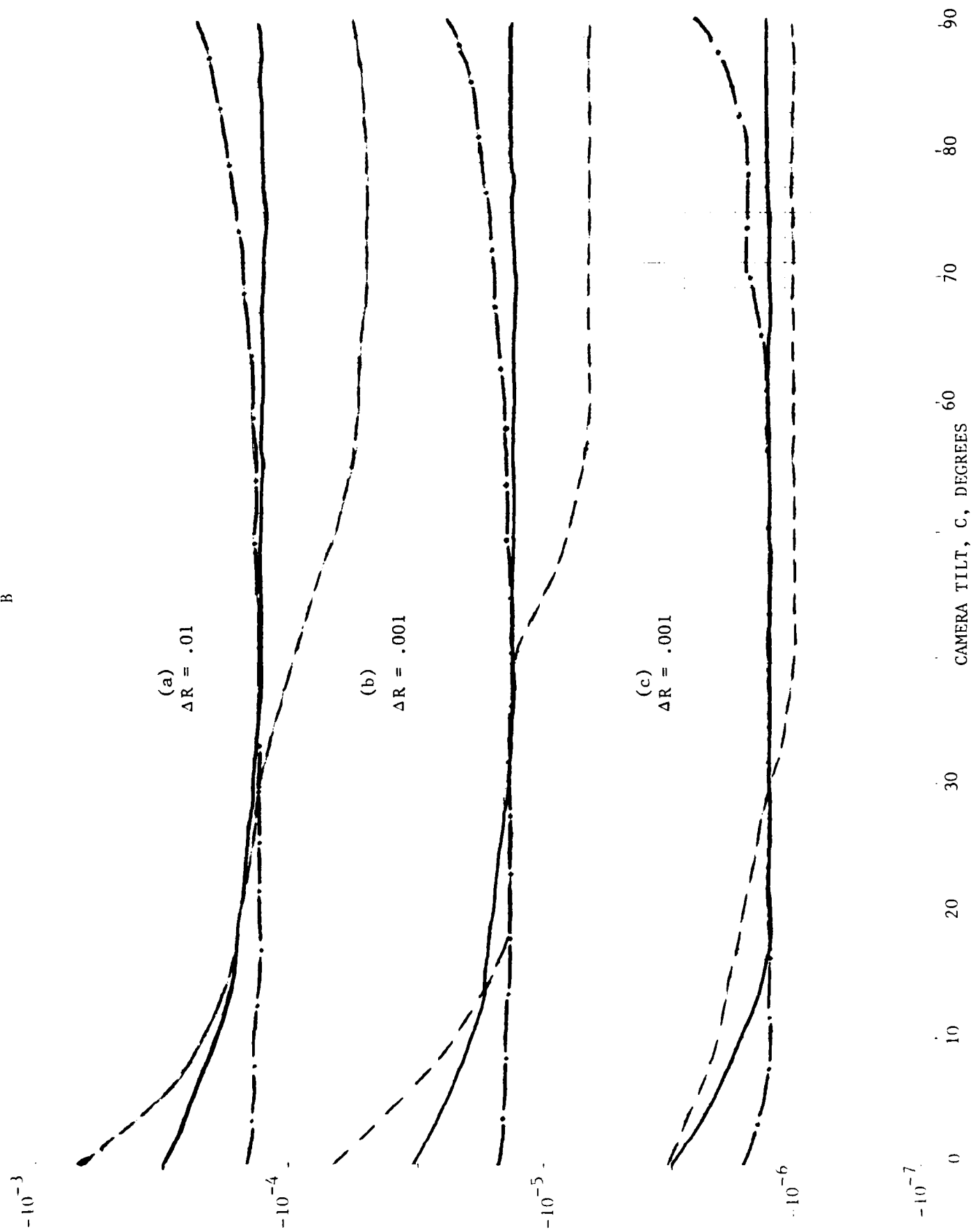


B.

The attached plot shows the changes detected in range ( $\Delta R'$ ) due to a change made in camera tilt (c) for the following parameters:

- a)  $\Delta R = .01$  a change in range  
Camera-projector distance (d) = 2  
Range (R) = 1 } red line  
Height of the object (y) = .5 }  
  
camera-project distance (d) = 1  
range of the object (R) = 1 } solid black line  
Height of the object (y) = .5 }  
  
d=1  
R=2 } dashed line  
y=.5 }  
  
b)  $\Delta R = .001$   
same as above  
  
c)  $\Delta R = .001$   
same as above

Therefore the plot shows the sensitivity of the computed range with respect to a change in range of the object versus the camera tilt. Note that the resolution (pixel size) in the CID is  $3\text{ }\mu\text{m}$ .



```

1
LIST
100 INPUT "ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=";D
110 INPUT "ENTER THE RANGE=";R
130 INPUT "ENTER THE CAMERA TILT=";C
150 INPUT "ENTER THE DESIRED CHANGE IN RANGE=";N
112 REM
133 PRINT "HEIGHT","DELTA R"
155 C1 = (C / 180) * (3.141592654)
116 REM
157 FOR Y = 0 TO D STEP 0.1
158 REM
159 REM
190 B1 = ATN (R / (D - Y))
200 A1 = - (R / ( SIN (B1))) * ( SIN (B1 - C1))
210 D1 = (R / ( SIN (B1))) * ( COS (B1 - C1))
215 F2 = .025
220 IM1 = (A1 / D1) * F2
221 V = IM1
231 REM
232 REM
240 N1 = N
292 M = 0
294 IF N < 0 THEN L = 1
295 IF N > 0 THEN L = 0
296 REM
297 REM
400 A2 = A1 * (( - 1) ^ (M + L + 1)) * (N1) * COS (C1)
410 D2 = D1 * (( - 1) ^ L) * (N1) * ( SIN (C1))
420 IM2 = (A2 / D2) * F2
431 REM
432 REM
435 W = IM2
440 T = W - V
451 REM
452 REM
440 IM3 = INT ((1 + .00000005) * 1000000) / 1000000
462 REM
464 REM
450 PRINT Y, IM3
461 NEXT Y
470 END

```

A

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR-1

ENTER THE RANGE=1

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT DELTA R'

0 -2E-06

.1 -2E-06

.2 -2E-06

.3 -2E-06

.4 -1E-06

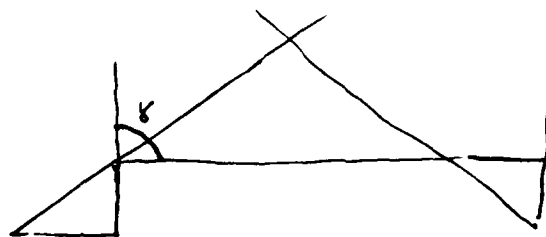
.5 -1E-06

.6 -1E-06

.7 -1E-06

.8 0

.9 0



ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.001

HEIGHT                      DELTA R'

0                            -2.5E-05

.1                           -2.2E-05

.2                           -2E-05

.3                           -1.7E-05

.4                           -1.5E-05

.5                           -1.2E-05

.6                           -1E-05

.7                           -7E-06

.8                           -5E-06

.9                           -2E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=3

ENTER THE RANGE=1

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.01

| HEIGHT | DELTA R'  |
|--------|-----------|
| 0      | -2.68E-04 |
| .1     | -2.23E-04 |
| .2     | -1.93E-04 |
| .3     | -1.73E-04 |
| .4     | -1.49E-04 |
| .5     | -1.24E-04 |
| .6     | -9.9E-05  |
| .7     | -7.4E-05  |
| .8     | -5E-05    |
| .9     | -2.5E-05  |

AD-A168 197

DEVELOPMENT OF ALGORITHMS AND HARDWARE FOR  
STRUCTURED-LIGHT VISION(U) GEORGE WASHINGTON UNIV  
WASHINGTON DC DEPT OF ELECTRICAL ENGIN..

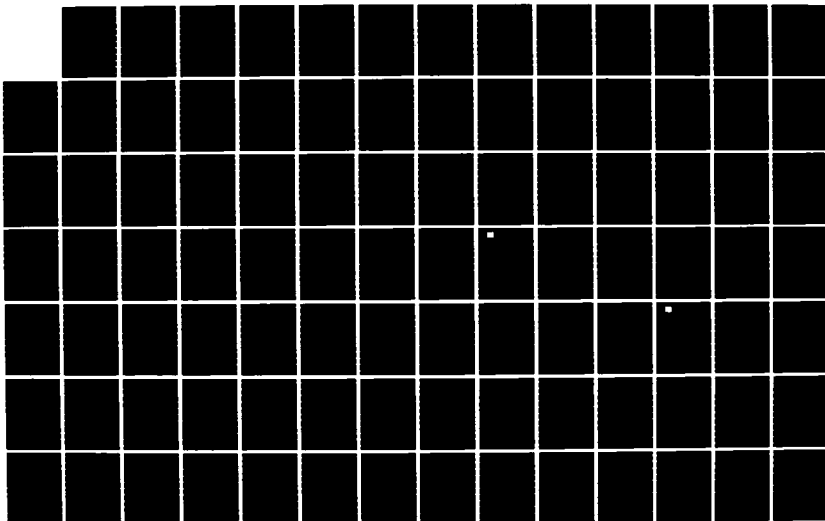
2/4

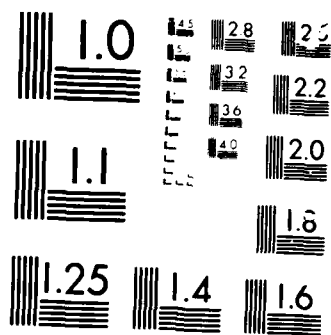
UNCLASSIFIED

M H LOEM ET AL. 1986 N00014-83-K-0578

F/G 28/6

NL





MICROSCOPE



\*\*\*\*\*

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE =1

ENTER THE CAMERA TILT=75

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT DELTA R'

0 -2E-06

.1 -2E-06

.2 -1E-06

.3 -1E-06

.4 -1E-06

.5 -1E-06

.6 -1E-06

.7 -1E-06

.8 0

.9 0

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTION :

ENTER THE RANGE=1

ENTER THE CAMERA FULI=75

ENTER THE DESIRED CHANGE IN RANGE=.001

HEIGHT                      DELTA R'

0                            -1.7E-05

.1                           -1.6E-05

.2                           -1.5E-05

.3                           -1.3E-05

.4                           -1.2E-05

.5                           -1E-05

.6                           -9E-06

.7                           -7E-06

.8                           -5E-06

.9                           -3E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE CAMERA TILT=75

ENTER THE DESIRED CHANGE IN RANGE=.01

| HEIGHT | DELTA K'  |
|--------|-----------|
| 0      | -1.65E-04 |
| .1     | -1.55E-04 |
| .2     | -1.44E-04 |
| .3     | -1.32E-04 |
| .4     | -1.18E-04 |
| .5     | -1.03E-04 |
| .6     | -8.7E-05  |
| .7     | -6.8E-05  |
| .8     | -4.8E-05  |
| .9     | -2.5E-05  |

IRUN

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE CAMERA TILT=45

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT                      DELTA K<sup>2</sup>

0                              -1E-06

.1                             -1E-06

.2                             -1E-06

.3                             -1E-06

.4                             -1E-06

.5                             -1E-06

.6                             -1E-06

.7                             -1E-06

.8                             -1E-06

.9                             0

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE CAMERA FILT=45

ENTER THE DESIRED CHANGE IN RANGE=.001

| HEIGHT | DELTA R <sup>2</sup> |
|--------|----------------------|
|--------|----------------------|

|   |          |
|---|----------|
| 0 | -1.2E-05 |
|---|----------|

|    |          |
|----|----------|
| .1 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .2 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .3 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .4 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .5 | -1.1E-05 |
|----|----------|

|    |        |
|----|--------|
| .6 | -1E-05 |
|----|--------|

|    |        |
|----|--------|
| .7 | -9E-06 |
|----|--------|

|    |        |
|----|--------|
| .8 | -7E-06 |
|----|--------|

|    |        |
|----|--------|
| .9 | -4E-06 |
|----|--------|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2  
 ENTER THE RANGE=1  
 ENTER THE CAMERA TILT=90  
 ENTER THE DESIRED CHANGE IN RANGE=.0001  
 HEIGHT                      DELTA K<sup>2</sup>  
 0                              -5E-06  
 .1                             -1E-06  
 .2                             -6E-06  
 .3                             -4E-06  
 .4                             -6E-06  
 .5                             -4E-06  
 .6                             -5E-06  
 .7                             -4E-06  
 .8                             -5E-06  
 .9                             -5E-06  
 1                              -2E-06  
 1.1                            -2E-06  
 1.2                            -2E-06  
 1.3                            -2E-06  
 1.4                            -1E-06  
 1.5                            -1E-06  
 1.6                            -1E-06  
 1.7                            -1E-06  
 1.8                            0  
 1.9                            0

ESTIMATE ERROR

ORUN

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.001

HEIGHT            DELTA R<sup>2</sup>

|     |          |
|-----|----------|
| 0   | -5E-05   |
| .1  | -4.7E-05 |
| .2  | -4.5E-05 |
| .3  | -4.2E-05 |
| .4  | -4E-05   |
| .5  | -3.7E-05 |
| .6  | -3.5E-05 |
| .7  | -3.2E-05 |
| .8  | -3E-05   |
| .9  | -2.7E-05 |
| 1   | -2.5E-05 |
| 1.1 | -2.2E-05 |
| 1.2 | -2E-05   |
| 1.3 | -1.7E-05 |
| 1.4 | -1.5E-05 |
| 1.5 | -1.2E-05 |
| 1.6 | -1E-05   |
| 1.7 | -7E-06   |
| 1.8 | -5E-06   |
| 1.9 | -2E-06   |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.01

| HEIGHT | DELTA R'  |
|--------|-----------|
| 0      | -4.93E-04 |
| .1     | -4.7E-04  |
| .2     | -4.46E-04 |
| .3     | -4.21E-04 |
| .4     | -3.96E-04 |
| .5     | -3.71E-04 |
| .6     | -3.47E-04 |
| .7     | -3.22E-04 |
| .8     | -2.97E-04 |
| .9     | -2.72E-04 |
| 1      | -2.48E-04 |
| 1.1    | -2.23E-04 |
| 1.2    | -1.98E-04 |
| 1.3    | -1.73E-04 |
| 1.4    | -1.49E-04 |
| 1.5    | -1.24E-04 |
| 1.6    | -9.9E-05  |
| 1.7    | -7.4E-05  |
| 1.8    | -5E-05    |
| 1.9    | -2.5E-05  |



ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE CAMERA TILT=75

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT                      DELTA K<sup>2</sup>

0                            -2E-06

.1                           -2E-06

.2                           -2E-06

.3                           -2E-06

.4                           -2E-06

.5                           -2E-06

.6                           -2E-06

.7                           -2E-06

.8                           -2E-06

.9                           -2E-06

1                           -2E-06

1.1                          -2E-06

1.2                          -1E-06

1.3                          -1E-06

1.4                          -1E-06

1.5                          -1E-06

1.6                          -1E-06

1.7                          -1E-06

1.8                          0

1.9                          0

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR--Z

ENTER THE RANGE=L

ENTER THE CAMERA FTL F=75

ENTER THE DESIRED CHANGE IN RANGE=.01

HEIGHT DELTA R<sup>2</sup>

|     |           |
|-----|-----------|
| 0   | -2.26E-04 |
| .1  | -2.22E-04 |
| .2  | -2.18E-04 |
| .3  | -2.14E-04 |
| .4  | -2.09E-04 |
| .5  | -2.03E-04 |
| .6  | -1.97E-04 |
| .7  | -1.9E-04  |
| .8  | -1.83E-04 |
| .9  | -1.74E-04 |
| 1   | -1.65E-04 |
| 1.1 | -1.55E-04 |
| 1.2 | -1.44E-04 |
| 1.3 | -1.32E-04 |
| 1.4 | -1.18E-04 |
| 1.5 | -1.03E-04 |
| 1.6 | -8.7E-05  |
| 1.7 | -6.8E-05  |
| 1.8 | -4.8E-05  |
| 1.9 | -2.5E-05  |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE CAMERA DUTY=75

ENTER THE DESIRED CHANGE IN RANGE=.1

| HEIGHT | DELTA R'   |
|--------|------------|
| 0      | -2.133E-03 |
| .1     | -2.097E-03 |
| .2     | -2.056E-03 |
| .3     | -2.012E-03 |
| .4     | -1.963E-03 |
| .5     | -1.909E-03 |
| .6     | -1.849E-03 |
| .7     | -1.784E-03 |
| .8     | -1.712E-03 |
| .9     | -1.632E-03 |
| 1      | -1.545E-03 |
| 1.1    | -1.449E-03 |
| 1.2    | -1.343E-03 |
| 1.3    | -1.227E-03 |
| 1.4    | -1.099E-03 |
| 1.5    | -9.57E-04  |
| 1.6    | -8.02E-04  |
| 1.7    | -6.3E-04   |
| 1.8    | -4.41E-04  |
| 1.9    | -2.32E-04  |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE CAMERA FDLT=45

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT DELTA R<sup>2</sup>

0 -1E-06

.1 -1E-06

.2 -1E-06

.3 -1E-06

.4 -1E-06

.5 -1E-06

.6 -1E-06

.7 -1E-06

.8 -1E-06

.9 -1E-06

1 -1E-06

1.1 -1E-06

1.2 -1E-06

1.3 -1E-06

1.4 -1E-06

1.5 -1E-06

1.6 -1E-06

1.7 -1E-06

1.8 -1E-06

1.9 0

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE CAMERA TILT=45

ENTER THE DESIRED CHANGE IN RANGE=.001

| HEIGHT | DELTA R <sup>2</sup> |
|--------|----------------------|
|--------|----------------------|

|   |          |
|---|----------|
| 0 | -1.1E-05 |
|---|----------|

|    |          |
|----|----------|
| .1 | -1.1E-05 |
|----|----------|

|    |          |
|----|----------|
| .2 | -1.1E-05 |
|----|----------|

|    |          |
|----|----------|
| .3 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .4 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .5 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .6 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .7 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .8 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| .9 | -1.2E-05 |
|----|----------|

|   |          |
|---|----------|
| 1 | -1.2E-05 |
|---|----------|

|     |          |
|-----|----------|
| 1.1 | -1.2E-05 |
|-----|----------|

|     |          |
|-----|----------|
| 1.2 | -1.2E-05 |
|-----|----------|

|     |          |
|-----|----------|
| 1.3 | -1.2E-05 |
|-----|----------|

|     |          |
|-----|----------|
| 1.4 | -1.2E-05 |
|-----|----------|

|     |          |
|-----|----------|
| 1.5 | -1.1E-05 |
|-----|----------|

|     |        |
|-----|--------|
| 1.6 | -1E-05 |
|-----|--------|

|     |        |
|-----|--------|
| 1.7 | -9E-06 |
|-----|--------|

|     |        |
|-----|--------|
| 1.8 | -7E-06 |
|-----|--------|

|     |        |
|-----|--------|
| 1.9 | -4E-06 |
|-----|--------|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2  
 ENTER THE RANGE=1  
 ENTER THE CAMERA TILT=45  
 ENTER THE DESIRED CHANGE IN RANGE=.1  
 HEIGHT                      DELTA R'

|     |            |
|-----|------------|
| 0   | -1.075E-03 |
| .1  | -1.092E-03 |
| .2  | -1.108E-03 |
| .3  | -1.124E-03 |
| .4  | -1.14E-03  |
| .5  | -1.154E-03 |
| .6  | -1.167E-03 |
| .7  | -1.178E-03 |
| .8  | -1.186E-03 |
| .9  | -1.19E-03  |
| 1   | -1.19E-03  |
| 1.1 | -1.184E-03 |
| 1.2 | -1.17E-03  |
| 1.3 | -1.144E-03 |
| 1.4 | -1.103E-03 |
| 1.5 | -1.042E-03 |
| 1.6 | -9.52E-04  |
| 1.7 | -8.24E-04  |
| 1.8 | -6.41E-04  |
| 1.9 | -3.79E-04  |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=2

ENTER THE CAMERA DILT=90

ENTER THE DESIRED CHANGE IN RANGE=.0001

| HEIGHT | DELTA R' |
|--------|----------|
|--------|----------|

|   |        |
|---|--------|
| 0 | -1E-06 |
|---|--------|

|    |        |
|----|--------|
| .1 | -1E-06 |
|----|--------|

|    |   |
|----|---|
| .2 | 0 |
|----|---|

|    |   |
|----|---|
| .3 | 0 |
|----|---|

|    |   |
|----|---|
| .4 | 0 |
|----|---|

|    |   |
|----|---|
| .5 | 0 |
|----|---|

|    |   |
|----|---|
| .6 | 0 |
|----|---|

|    |   |
|----|---|
| .7 | 0 |
|----|---|

|    |   |
|----|---|
| .8 | 0 |
|----|---|

|    |   |
|----|---|
| .9 | 0 |
|----|---|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=.2

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.1

| HEIGHT | DELTA R <sup>2</sup> |
|--------|----------------------|
| 0      | -3.95E-04            |
| .1     | -4.36E-04            |
| .2     | -4.76E-04            |
| .3     | -4.17E-04            |
| .4     | -3.57E-04            |
| .5     | -2.98E-04            |
| .6     | -2.38E-04            |
| .7     | -1.79E-04            |
| .8     | -1.19E-04            |
| .9     | -6E-05               |



ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=2

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.01

HEIGHT DELTA R<sup>2</sup>

0 -6.0E-05

.1 -5.6E-05

.2 -5E-05

.3 -4.4E-05

.4 -3.7E-05

.5 -3.1E-05

.6 -2.5E-05

.7 -1.9E-05

.8 -1.2E-05

.9 -4E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR-Z

ENTER THE RANGE=3

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT DELTA K<sup>2</sup>

0 -1E-06

.1 -1E-06

.2 0

.3 0

.4 0

.5 0

.6 0

.7 0

.8 0

.9 0

1 0

1.1 0

1.2 0

1.3 0

1.4 0

1.5 0

1.6 0

1.7 0

1.8 0

1.9 0

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=3

ENTER THE CAMERA TILT=90

ENTER THE DESIRED CHANGE IN RANGE=.01

HEIGHT                      DELTA R'

0                            -5.5E-05

.1                           -6.3E-05

.2                           -5E-05

.3                           -4.7E-05

.4                           -4.4E-05

.5                           -4.2E-05

.6                           -3.9E-05

.7                           -3.6E-05

.8                           -3.3E-05

.9                           -3E-05

1                           -2.8E-05

1.1                          -2.5E-05

1.2                          -2.2E-05

1.3                          -1.9E-05

1.4                          -1.7E-05

1.5                          -1.4E-05

1.6                          -1.1E-05

1.7                          -8E-06

1.8                          -6E-06

1.9                          -3E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=0

ENTER THE CAMERA TILT=75

ENTER THE DESIRED CHANGE IN RANGE=.0001

HEIGHT                      DELTA R'

|    |   |
|----|---|
| 0  | 0 |
| .1 | 0 |
| .2 | 0 |
| .3 | 0 |
| .4 | 0 |
| .5 | 0 |
| .6 | 0 |
| .7 | 0 |
| .8 | 0 |
| .9 | 0 |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=5

ENTER THE CAMERA TILT=25

ENTER THE DESIRED CHANGE IN RANGE=.01

HEIGHT                      DELTA K<sup>2</sup>

0                            -1E-05

.1                           -9E-06

.2                           -8E-06

.3                           -7E-06

.4                           -6E-06

.5                           -5E-06

.6                           -4E-06

.7                           -3E-06

.8                           -2E-06

.9                           -1E-06

1  
11151

100 INPUT "ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=";D  
110 INPUT "ENTER THE RANGE=";R  
120 INPUT "ENTER THE HEIGHT=";Y  
130 INPUT "ENTER THE DESIRED CHANGE IN RANGE=";N

132 REM  
133 PRINT "CAMERA TILT", "DELTA R"  
134 FOR C = 0 TO 90 STEP 10  
135 C1 = (C / 180) \* (3.141592654)  
136 REM  
137 REM  
138 REM  
139 REM  
140 B1 = ATN (R / (D - Y))  
200 A1 = - (R / ( SIN (B1))) \* ( SIN (B1 - C1))  
210 D1 = (R / ( SIN (B1))) \* ( COS (B1 - C1))  
215 F2 = .025  
220 IM1 = (A1 / D1) \* F2  
221 V = IM1  
231 REM  
232 REM  
240 N1 = N  
252 N = 0  
254 IF N < 0 THEN L = 1  
255 IF N >= 0 THEN L = 0  
256 REM  
257 REM  
300 A2 = A1 + ((- 1) ^ (M + L + 1)) \* (N1) \* COS (C1)  
310 D2 = D1 + ((- 1) ^ L) \* (N1) \* ( SIN (C1))  
320 IM2 = (A2 / D2) \* F2  
321 REM  
322 REM  
325 W = IM2  
330 T = W - V  
331 REM  
332 REM  
340 IM3 = INT ((T + .00000005) \* 1000000) / 1000000  
352 REM  
354 REM  
355 PRINT C, IM3  
365 NEXT C  
370 END

B

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE = .0001

CAMERA TILT            DELTA R<sup>2</sup>

0                      -5E-06

10                     -3E-06

20                     -2E-06

30                     -1E-06

40                     -1E-06

50                     -1E-06

60                     -1E-06

70                     -1E-06

80                     -1E-06

90                     -1E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.10

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -5E-05               |
| 10          | -2.8E-05             |
| 20          | -1.9E-05             |
| 30          | -1.4E-05             |
| 40          | -1.2E-05             |
| 50          | -1.1E-05             |
| 60          | -1E-05               |
| 70          | -1E-05               |
| 80          | -1.1E-05             |
| 90          | -1.2E-05             |



ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=.1

ENTER THE HEIGHT=.1

ENTER THE DESIRED CHANGE IN RANGE=.01

| CAMERA TILT | DELTA K <sup>2</sup> |
|-------------|----------------------|
| 0           | -5E-04               |
| 10          | -2.81E-04            |
| 20          | -1.89E-04            |
| 30          | -1.43E-04            |
| 40          | -1.18E-04            |
| 50          | -1.05E-04            |
| 60          | -1E-04               |
| 70          | -1E-04               |
| 80          | -1.08E-04            |
| 90          | -1.24E-04            |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE HEIGHT=10

ENTER THE DESIRED CHANGE IN RANGE=.0001

CAMERA TILT            DELTA R<sup>2</sup>

0                      -2E-06

10                     -1E-06

20                     -1E-06

30                     -1E-06

40                     -1E-06

50                     -1E-06

60                     -1E-06

70                     -1E-06

80                     -2E-06

90                     -4E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
|-------------|----------------------|

|   |          |
|---|----------|
| 0 | -1.7E-05 |
|---|----------|

|    |          |
|----|----------|
| 10 | -1.4E-05 |
|----|----------|

|    |          |
|----|----------|
| 20 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| 30 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| 40 | -1.2E-05 |
|----|----------|

|    |          |
|----|----------|
| 50 | -1.3E-05 |
|----|----------|

|    |          |
|----|----------|
| 60 | -1.4E-05 |
|----|----------|

|    |          |
|----|----------|
| 70 | -1.8E-05 |
|----|----------|

|    |          |
|----|----------|
| 80 | -2.6E-05 |
|----|----------|

|    |          |
|----|----------|
| 90 | -3.7E-05 |
|----|----------|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1

ENTER THE HEIGHT=.1

ENTER THE DESIRED CHANGE IN RANGE=.01

| CAMERA IDLE | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -1.67E-04            |
| 10          | -1.37E-04            |
| 20          | -1.22E-04            |
| 30          | -1.16E-04            |
| 40          | -1.13E-04            |
| 50          | -1.25E-04            |
| 60          | -1.43E-04            |
| 70          | -1.77E-04            |
| 80          | -2.4E-04             |
| 90          | -3.71E-04            |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.7

ENTER THE DESIRED CHANGE IN RANGE=.0001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
|-------------|----------------------|

|   |        |
|---|--------|
| 0 | -3E-06 |
|---|--------|

|    |        |
|----|--------|
| 10 | -3E-06 |
|----|--------|

|    |        |
|----|--------|
| 20 | -2E-06 |
|----|--------|

|    |        |
|----|--------|
| 30 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 40 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 50 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 60 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 70 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 80 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 90 | -1E-06 |
|----|--------|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=.1

ENTER THE HEIGHT=.7

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA DLT | DELTA R <sup>2</sup> |
|------------|----------------------|
|------------|----------------------|

|   |          |
|---|----------|
| 0 | -8.5E-05 |
|---|----------|

|    |          |
|----|----------|
| 10 | -3.4E-05 |
|----|----------|

|    |          |
|----|----------|
| 20 | -1.9E-05 |
|----|----------|

|    |          |
|----|----------|
| 30 | -1.3E-05 |
|----|----------|

|    |        |
|----|--------|
| 40 | -1E-05 |
|----|--------|

|    |        |
|----|--------|
| 50 | -8E-06 |
|----|--------|

|    |        |
|----|--------|
| 60 | -7E-06 |
|----|--------|

|    |        |
|----|--------|
| 70 | -7E-06 |
|----|--------|

|    |        |
|----|--------|
| 80 | -7E-06 |
|----|--------|

|    |        |
|----|--------|
| 90 | -7E-06 |
|----|--------|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.7

ENTER THE DESIRED CHANGE CN RANGE=.01

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -8.33E-04            |
| 10          | -3.4E-04             |
| 20          | -1.92E-04            |
| 30          | -1.29E-04            |
| 40          | -9.8E-05             |
| 50          | -8.1E-05             |
| 60          | -7.2E-05             |
| 70          | -6.8E-05             |
| 80          | -6.9E-05             |
| 90          | -7.4E-05             |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.3

ENTER THE DESIRED CHANGE IN RANGE=.0001

| CAMERA TILT | DELTA R' |
|-------------|----------|
|-------------|----------|

|   |        |
|---|--------|
| 0 | -4E-06 |
|---|--------|

|    |        |
|----|--------|
| 10 | -2E-06 |
|----|--------|

|    |        |
|----|--------|
| 20 | -2E-06 |
|----|--------|

|    |        |
|----|--------|
| 30 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 40 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 50 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 60 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 70 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 80 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 90 | -2E-06 |
|----|--------|



ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.3

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -3.6E-05             |
| 10          | -2.3E-05             |
| 20          | -1.8E-05             |
| 30          | -1.4E-05             |
| 40          | -1.3E-05             |
| 50          | -1.2E-05             |
| 60          | -1.2E-05             |
| 70          | -1.3E-05             |
| 80          | -1.4E-05             |
| 90          | -1.7E-05             |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.8

ENTER THE DESIRED CHANGE IN RANGE=.0001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -1.2E-05             |
| 10          | -4E-06               |
| 20          | -2E-06               |
| 30          | -1E-06               |
| 40          | -1E-06               |
| 50          | -1E-06               |
| 60          | -1E-06               |
| 70          | 0                    |
| 80          | 0                    |
| 90          | 0                    |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=1

ENTER THE HEIGHT=.9

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -2.5E-04             |
| 10          | -3.4E-05             |
| 20          | -1.3E-05             |
| 30          | -7E-06               |
| 40          | -5E-06               |
| 50          | -4E-06               |
| 60          | -3E-06               |
| 70          | -3E-06               |
| 80          | -2E-06               |
| 90          | -2E-06               |

SYNTAX ERROR

IRUN

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=2

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.0001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
|-------------|----------------------|

|   |        |
|---|--------|
| 0 | -5E-06 |
|---|--------|

|    |        |
|----|--------|
| 10 | -2E-06 |
|----|--------|

|    |        |
|----|--------|
| 20 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 30 | -1E-06 |
|----|--------|

|    |   |
|----|---|
| 40 | 0 |
|----|---|

|    |   |
|----|---|
| 50 | 0 |
|----|---|

|    |   |
|----|---|
| 60 | 0 |
|----|---|

|    |   |
|----|---|
| 70 | 0 |
|----|---|

|    |   |
|----|---|
| 80 | 0 |
|----|---|

|    |   |
|----|---|
| 90 | 0 |
|----|---|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=2

ENTER THE HEIGHT=.1

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R' |
|-------------|----------|
| 0           | -5E-05   |
| 10          | -1.8E-05 |
| 20          | -9E-06   |
| 30          | -6E-06   |
| 40          | -6E-06   |
| 50          | -4E-06   |
| 60          | -5E-06   |
| 70          | -3E-06   |
| 80          | -5E-06   |
| 90          | -3E-06   |

SYNTAX ERROR

IRON

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=2

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.01

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -5E-04               |
| 10          | -1.77E-04            |
| 20          | -9.4E-05             |
| 30          | -6.1E-05             |
| 40          | -4.5E-05             |
| 50          | -3.6E-05             |
| 60          | -3.2E-05             |
| 70          | -3E-05               |
| 80          | -2.9E-05             |
| 90          | -3.1E-05             |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=5

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.0001

CAMERA TILT            DELTA R<sup>2</sup>

0                      -5E-06

10                     -1E-06

20                     -1E-06

30                     0

40                     0

50                     0

60                     0

70                     0

80                     0

90                     0

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=5

ENTER THE HEIGHT=.10

ENTER THE DESIRED CHANGE IN RANGE=.001

CAMERA LIFT            DELTA R<sup>2</sup>

|    |          |
|----|----------|
| 0  | -5E-05   |
| 10 | -1.2E-05 |
| 20 | -6E-06   |
| 30 | -3E-06   |
| 40 | -2E-06   |
| 50 | -2E-06   |
| 60 | -2E-06   |
| 70 | -1E-06   |
| 80 | -1E-06   |
| 90 | -1E-06   |



ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=3

ENTER THE HEIGHT=7.0

EXTRA (IGNORE)

ENTER THE DESIRED CHANGE IN RANGE=.01

| CAMERA TILT | DELTA R' |
|-------------|----------|
| 0           | -2.5E-04 |
| 10          | -1.1E-04 |
| 20          | -6.5E-05 |
| 30          | -4.5E-05 |
| 40          | -3.4E-05 |
| 50          | -2.9E-05 |
| 60          | -2.6E-05 |
| 70          | -2.5E-05 |
| 80          | -2.5E-05 |
| 90          | -2.8E-05 |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=5

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.0001

| CAMERA TILT | DELTA R' |
|-------------|----------|
|-------------|----------|

|   |        |
|---|--------|
| 0 | -5E-06 |
|---|--------|

|    |        |
|----|--------|
| 10 | -1E-06 |
|----|--------|

|    |   |
|----|---|
| 20 | 0 |
|----|---|

|    |   |
|----|---|
| 30 | 0 |
|----|---|

|    |   |
|----|---|
| 40 | 0 |
|----|---|

|    |   |
|----|---|
| 50 | 0 |
|----|---|

|    |   |
|----|---|
| 60 | 0 |
|----|---|

|    |   |
|----|---|
| 70 | 0 |
|----|---|

|    |   |
|----|---|
| 80 | 0 |
|----|---|

|    |   |
|----|---|
| 90 | 0 |
|----|---|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=.5

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R' |
|-------------|----------|
|-------------|----------|

|   |        |
|---|--------|
| 0 | -5E-05 |
|---|--------|

|    |        |
|----|--------|
| 10 | -7E-06 |
|----|--------|

|    |        |
|----|--------|
| 20 | -3E-06 |
|----|--------|

|    |        |
|----|--------|
| 30 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 40 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 50 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 60 | -1E-06 |
|----|--------|

|    |        |
|----|--------|
| 70 | -1E-06 |
|----|--------|

|    |   |
|----|---|
| 80 | 0 |
|----|---|

|    |   |
|----|---|
| 90 | 0 |
|----|---|

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=1

ENTER THE RANGE=.5

ENTER THE HEIGHT=.5

ENTER THE DESIRED CHANGE IN RANGE=.01

CAMERA TILT            DELTA R<sup>2</sup>

0                      -5E-04

10                     -6.7E-05

20                     -2.6E-05

30                     -1.5E-05

40                     -1E-05

50                     -7E-06

60                     -6E-06

70                     -5E-06

80                     -5E-06

90                     -5E-06

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2  
ENTER THE RANGE=1.0  
ENTER THE HEIGHT=1  
ENTER THE DESIRED CHANGE IN RANGE=.0001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -2E-06               |
| 10          | -2E-06               |
| 20          | -1E-06               |
| 30          | -1E-06               |
| 40          | -1E-06               |
| 50          | -1E-06               |
| 60          | -1E-06               |
| 70          | -1E-06               |
| 80          | -1E-06               |
| 90          | -1E-06               |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=1.5

ENTER THE HEIGHT=1

ENTER THE DESIRED CHANGE IN RANGE=.001

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -2.5E-05             |
| 10          | -1.6E-05             |
| 20          | -1.2E-05             |
| 30          | -1E-05               |
| 40          | -8E-06               |
| 50          | -6E-06               |
| 60          | -5E-06               |
| 70          | -4E-06               |
| 80          | -3E-06               |
| 90          | -1.1E-05             |

ENTER THE DISTANCE BETWEEN CAMERA AND PROJECTOR=2

ENTER THE RANGE=0.5

ENTER THE HEIGHT=0

ENTER THE DESIRED CHANGE IN RANGE=.01

| CAMERA TILT | DELTA R <sup>2</sup> |
|-------------|----------------------|
| 0           | -2.5E-04             |
| 10          | -1.61E-04            |
| 20          | -1.18E-04            |
| 30          | -9.5E-05             |
| 40          | -8.3E-05             |
| 50          | -7.8E-05             |
| 60          | -7.7E-05             |
| 70          | -8.1E-05             |
| 80          | -9.1E-05             |
| 90          | -1.1E-04             |

APPENDIX C

DAD BOX DESCRIPTION





# DIGITAL/ANALOG DESIGN

ELECTRONIC DESIGN - PRODUCT DEVELOPMENT

---

530 BROADWAY  
NEW YORK, N.Y. 10012  
(212) 966-0410

## DIGITAL VIDEO DEVELOPMENT SYSTEM (DVDS) MODEL 7 OPERATORS MANUAL

FOR: NATIONAL BUREAU OF STANDARDS  
PROGRAMMABLE AUTOMATION DIVISION

11/15/83

Copyright 1983 Digital/Analog Design Associates, Inc.

## TABLE OF CONTENTS

### 1.0 System Introduction

- 1.1 Controller
- 1.2 Image Memories
  - 1.2.1 Reference Memory
  - 1.2.2 Bit Plane Memory
- 1.3 Address Generator
- 1.4 Analog Input/Output
- 1.5 Computer Interface
- 1.6 Case and Electrical

### 2.0 Computer Interface

- 2.1 Description
- 2.2 List of Registers, Addresses, and Modes
  - 2.2.1 Computer Interface
  - 2.2.2 Mapping RAM
  - 2.2.3 Bit Plane Memory
  - 2.2.4 Operating Modes
- 2.3 Direct Memory Access (DMA)
  - 2.3.1 Introduction
  - 2.3.2 DMA Read
  - 2.3.3 DMA Write (Reference Memory only)
- 2.4 Interfacing Requirements for the Host Computer
  - 2.4.1 Computer Output Ports
  - 2.4.2 Computer Input Port

### 3.0 Analog Input/Output

- 3.1 Description
- 3.2 Analog to Digital Section
- 3.3 Digital to Analog Section
- 3.4 Composite Sync.
- 3.5 Camera Requirements
- 3.6 Adjustment Procedure

### 4.0 Mapping RAM

- 4.1 Description
- 4.2 Loading the Mapping RAM

### 5.0 Address Generator

- 5.1 Description

### 6.0 Computer Interface Connector List

## 1.0 System Introduction

The DVDS Model 7 with Comparator/Slicer Option can be used for adaptive thresholding of real-time video images. A full video field can be stored in the reference memory to 8 bit greyscale resolution. The stored image can then be compared with the real-time video input and the resulting one bit image stored in the other, bit plane, memory. Eight different fields may be processed and stored in this way. The bit plane memory may be displayed directly or through a 256 x 8 bit Mapping RAM which implements an arbitrary programmable function. In addition, the host computer can load or unload the reference memory and unload the bit plane memory either directly or through the Mapping RAM.

### 1.1 Controller

The controller card contains the master clock oscillator, PROMs with all the necessary timing and sync signals, and the circuitry needed for controlling the dynamic RAMs used in the image memories.

### 1.2 Image Memories

There are two memory cards, the reference and the bit plane. Each memory card uses 16K dynamic RAMs arranged 256 H by 240 V by 8 deep. They run in real time, synchronous with the video display. Since the RAMs do not access fast enough, they are multiplexed 4 deep. For a read cycle, four pixels are addressed in parallel and are serially shifted out. The write cycle works similarly but in the opposite direction.

#### 1.2.1 Reference Memory

The reference memory stores an eight bit gray scale image. It may be filled with data from the video input or directly written into through the DMA interface from the host computer. It may be displayed through the video output or unloaded through the DMA interface to the computer.

### 1.2.2 Bit Plane Memory

The bit plane memory is configured as eight 256 X 240 x 1 bit planes. Input data to any of the planes comes from a magnitude comparator. This compares the live video input from the A/D with the image stored in the reference memory on a pixel by pixel basis. An offset may be specified which is added to the pixel values from the reference memory. This provides noise immunity when using the reference memory to remove differential scene lighting. If the input pixel is less than or equal to the reference pixel (plus offset), a zero is stored. If it is greater, a one is stored. When the reference pixel value plus offset exceeds 255 the value stored in the bit plane is automatically zero since the input pixel must be less than 256. Only one plane may be written into at a time. This memory may be displayed through the video output or unloaded to the computer through the DMA interface. The RAM look-up table may be placed between the bit plane memory and either of these output channels (see 1.5).

### 1.3 Address Generator

This card generates the addresses used to read from and write to the memory in the video modes. It also generates the refresh addresses for the dynamic RAMs. Its addressing modes are fixed.

### 1.4 Analog Input/Output

This card is the interface from the camera to the frame store and from the frame store to the CRT monitor. The video input from the camera is low-pass filtered, amplified, and digitized at 5 MHz to 8 bit gray scale resolution. The digitized video is then put onto the system bus destined for the memory. For video output there is a selector that allows the viewing of any of three sources, the reference memory, the bit plane memory and the bit plane memory through the Mapping RAM. With any of these sources either the entire gray scale or any of the eight individual planes may be displayed. The translation from digital to analog is done at 5 MHz in an 8 bit converter.

### 1.5 Computer Interface

The computer interface card is the link between the frame store and the host computer. It uses an 8 bit bidirectional data bus and an 8 bit address bus. This card allows the user to program the frame store into its various modes. It also has its own address generator for the image memories that allows bidirectional Direct Memory Access (DMA). A 256 x 8 bit RAM look-up table is included for arbitrary translation of bit plane memory data. This can operate on data output to the DAC or through the DMA channel.

## 1.6 Case and Electrical

The Frame store is packaged in an S-100 type computer case. The motherboard has been modified by adding an A.C. termination on each line. It uses standard S-100 wire wrap cards. There is an internal power supply that uses 120 VAC  $\pm$  10%. The on/off switch has a key lock and is located, along with the power indicator LED, on the front panel. Due to several internal reset circuits, after turning the unit off, the user should wait 10 seconds before turning the unit back on. All video connections are made with BNC connectors located on the back panel. The computer interface is via a 37 pin subminiature D type connector on the back panel. The fuseholder is also on the back panel. The fuse is a 3 amp slo-blo type.

## 2.0 Computer Interface

### 2.1 Description

The DVDS is designed to be a peripheral to a host computer. Its modes of operation are programmable and once programmed it needs no further intervention to operate. Programming is done by loading registers in the DVDS that control specific functions.

Two 8 bit output ports must be available from the computer. One, the command port, specifies which register is to be loaded. The other, the output data port, carries the data to be loaded. See paragraph 2.2 for the list of registers. These registers are write only, that is they cannot be read. Because the DVDS uses the handshaking line on the command port to initiate register loading, the data port must be loaded before the command port. Since the registers may come up at random on power up, they should be initialized to a known condition before the DVDS is used. This is called booting the system. See the program listing "BOOT". An output port from the DVDS is used to unload the image memory to the host computer.

## 2.2 List of Registers, Addresses, and Modes

### 2.2.1 Computer Interface

#### Address

- 0      Bit 0 : 0 = register loading during vertical drive only  
              1 = register loading during entire frame  
            All other bits are "don't care"
- 1      Bit 0 : 0 = Reference Memory Live  
              1 = Reference Memory Frozen
- Bit 1 : 0 = Bit Plane Memory Frozen  
                  1 = Bit Plane Memory Live
- Bit 2 : 0 = DMA Output Mode  
                  1 = DMA Input Mode
- Bit 3 : 0 = Normal (Video IN, OUT) Mode  
                  1 = DMA Mode  
            Note: In the DMA mode both memories are  
                  automatically prevented from writing in  
                  data from the A/D and the reference memory  
                  is displayed irregardless of any other  
                  controls.  
            All other bits are "don't care"
- 2      DMA Horizontal Address Register    (0-255)
- 3      DMA Vertical Address Register    (0-255)
- 4      DMA enable. During a read this is a flag, that is, it  
            initiates the read function, during write it contains  
            input pixel data 0-255.
- 5      Bit 0 : 0 = Display Reference Memory  
              1 = Display Bit Plane Memory
- Bit 1 : 0 = Direct Memory Display  
                  1 = Display Bit Plane Memory through Mapping RAM
- Bit 2 : 0 = Display Full grey scale  
                  1 = Display Single Bit of selected Memory  
            Note: The Single Bit display of either the  
                  Reference Memory, the Bit Plane Memory or the  
                  Bit Plane through the Mapping RAM may be chosen.  
            The bit is chosen by Register 6.

Bit 3 : 0 = A/D on Input Data Bus  
1 = Command Data Bus on Input Data Bus  
Note: This allows Data from the Host Computer to be placed on the Bit Plane Memory Input rather than the Input Video. This is useful for testing.

All other Bits are "don't care"

6 Bits 0-2 : Writing into the Bit Plane Memory occurs one plane at a time. These three bits determine which plane (0-7) is active (live). For the display of a single bit of one of the output sources (Register 5 Bit 2 High) register 6 selects the bit to be displayed.

All other Bits are "don't care"

## 2.2.2 Mapping RAM

### Address

224 Bit 0 : 0 = Normal Operation  
1 = Load Masking RAMs  
225 write enable for Mapping RAM (0-255)  
227 write address register (0-255)

## 2.2.3 Bit Plane Memory

### Address

255 Bit Plane Memory Offset Register (0-255)

Note: This adds a selectable bias to the data from the reference memory before being compared with data from the A/D for the bit plane memory. This is useful for removing the effects of noise.

#### 2.2.4 Operating Modes

|     | Mode   | Register 1 | Register 5 |
|-----|--|------------|------------|
| 1.  | Both Memories Frozen,<br>Reference Memory Displayed.   | 1          | 0          |
| 2.  | Reference Memory Live, Bit<br>Plane Memory Frozen, Reference<br>Memory Displayed.  | 0          | 0          |
| 3.  | Both Memories Frozen,<br>Bit Plane Memory Displayed.   | 1          | 1          |
| 4.  | Bit Plane Memory Live, Reference<br>Memory Frozen, Bit Plane Memory<br>Displayed, Register 6 contains the<br>Plane Number which is Live (0-7). | 3          | 1          |
| 5.  | Both Memories Frozen,<br>Display Bit Plane Memory through<br>Mapping Memory  | 1          | 3          |
| 6.  | Bit Plane Memory Live,<br>Display Bit Plane Memory through<br>Mapping RAM. Register 6<br>contains the Plane Number which<br>is Live (0-7).     | 3          | 3          |
| 7.  | Both Memories Frozen, Reference<br>Memory Displayed, DMA Write into<br>Reference Memory.   | 8          | 0          |
| 8.  | Both Memories Frozen, Reference<br>Memory Displayed, DMA read from<br>Reference Memory.  | 13         | 0          |
| 9.  | Both Memories Frozen, Bit Plane<br>Memory Displayed, DMA read from<br>Bit Plane Memory.  | 13         | 1          |
| 10. | Both Memories Frozen, Bit Plane,<br>DMA read from Bit Plane Memory<br>through the Mapping RAM.   | 13         | 3          |

Note: In any of these modes a single plane of the output source may be viewed by adding 4 to the number in Register 5 and loading Register 6 with the Plane number (0-7)



## 2.3 Direct Memory Access (DMA)

### 2.3.1 Introduction

The DVDS allows the host computer to directly access the memories. Frozen images from either memory or from the bit plane memory through the Mapping RAM may be unloaded to the computer via the high speed DMA interface. Additionally, the reference memory can be directly loaded from the computer. The transfer rate in either direction can exceed 200K bytes/second. A DMA address generator is used for both read and write operations. Pixel coordinates may be loaded into the horizontal and vertical address registers for random access of the image memory. Additionally, the DMA address generator will autoincrement after each read or write operation. This allows a starting address to be specified and then the circuit will provide sequential addresses itself. This speeds the data transfer rate substantially. The top left pixel is at address 0 H, 0 V. The top right pixel is at address 255 H, 0 V. The bottom left pixel is at address 0 H, 240 V. The bottom right pixel is at location 255 H, 240 V. The addresses will autoincrement from the last address loaded into the horizontal address register until 255 is reached. Then the vertical address will increment by one and the horizontal address overflows back to 0.

### 2.3.2 DMA Read

The algorithm for performing DMA read is outlined below. See also the program listings for REF.READ and MOVLOD.OBJ0

1. load register 0 with 1 - allows DMA access during entire frame.
2. load register 1 with 13 - this freezes the images and prepares for a DMA read operation.
3. load register 5 with the proper code for the desired output source (0 = Reference Memory, 1 = Bit Plane Memory, 3 = Bit Plane Memory through the Mapping RAM).
4. load register 2 (DMA horizontal address register) with H pixel coordinate.
5. load register 3 (DMA vertical address register) with V pixel coordinate.
6. load register 4 to assert the DMA read at the previously specified address. The pixel will be sent to the computer via the DVDS output port.

7. for random access go to 4. for sequential access go to 6. and repeat until the desired number of pixels are recieved.
8. load register 1 with 1. - this returns the system to its normal state and keeps the images frozen.

Note: The handshaking lines should be checked after each step

### 2.3.3 DMA Write (Reference Memory Only)

The proceedure for performing DMA write is similar to the DMA read. Also see the program listing REF.RAMP and MOVLOD.OBJO

1. load register 0 with 1 - this allows DMA access during the entire frame.
2. load register 1 with 9. - this freezes the image and prepares the unit for write.
3. load register 5 with 0 - this displays the reference memory
4. load register 2 with the pixel H coordinate.
5. load register 3 with the pixel V coordinate.
6. load register 4 with the pixel to be written into the image memory.
7. for random access go to 4. for sequential access go to 6. and repeat until the desired number of pixels are transferred.
8. load register 1 with 1 - this returns the unit to normal operation with the images frozen.

Note: The handshaking line should be checked after each step

## 2.4 Interfacing Requirements for the Computer

### 2.4.1 Computer Output Ports

Two output ports must be provided by the computer, the command port and the data port. Both must be 8 bits wide, active high at normal TTL levels. All lines are buffered in the DVDS and present 1  $\Omega$  LS TTL load. 560 ohm pull up resistors to +5 volts are included on these lines in the DVDS. These ports are treated by the unit as one 16 bit port. If separate ports are used it is important to load the data port before the command port, which has the handshaking lines associated with it. Once data has been set up and latched on both ports the COMPUTER READY line should go high. The DVDS will latch both ports 300 ns. after the rising edge of this signal. COMPUTER READY should be a standard TTL level signal. At this time the DEVICE BUSY line will go High to indicate that the command is being executed. The value loaded into register 0 determines when commands will be executed. In the "V.Drive Only" mode (0), a command will be held until the next vertical blanking interval before execution. This is useful for freezing the memories at the end of a field. Other commands such as DMA transfers should take place at high speed so register 0 is put into the "Anytime" mode (1). Once the DVDS has executed the command from the host computer, the DEVICE BUSY line (TTL) will go low. Also at this time the BUSY strobe will pulse for 1 microsecond. These indicate that the computer may send another command. Any commands sent before this time will be lost.

### 2.4.2 Computer Input Port

One 8 bit input port to the computer is required to read data from the image memory. This port must be standard level, TTL, active high. After a DMA READ command has been executed, the pixel datum will be placed on the computer input port. The DEVICE READY line will then pulse for 1 microsecond to strobe this information into the computer input port. This signal is at standard TTL levels. Note that the data should be strobed in on the trailing edge of the DEVICE READY line.

### 3.0 Analog Input/Output

#### 3.1 Description

This card accepts the video input from a camera, filters the signal, digitizes it, and passes it to the image memory. It also receives digital video from various sources, converts it to analog video, and passes it through a 75 ohm driver to the CRT monitor. In addition, composite sync is available for driving the camera. BNC connectors for video in, video out, and composite sync are located on the back of the DVDS.

#### 3.2 Analog to Digital section

This section accepts the input video from the camera and digitizes it.

|                       |                          |
|-----------------------|--------------------------|
| input impedance       | 75 ohms                  |
| input signal level    | 1 volt sync tip to white |
| sample rate           | 5 MHz                    |
| gray level resolution | 256 levels               |
| number of bits        | 8                        |
| filter cutoff         | 2 MHz, 24 db/octave      |

A white clip light is provided on the front panel. This will come on when the video signal is too high, causing the analog to digital converter to overflow.

#### 3.3 Digital to Analog section

Either of the stored images or the arithmetic unit may be selected for display. In addition to the full eight bit grey scale display, a single bit plane of any of these sources may be selected for viewing with the bit plane address register (see 2.2.1). The single bit mode output is either full black or full white. Blanking and sync are added digitally so there is no adjustment of sync or set up levels. The composite signal is filtered to remove spurious high frequency components.

|                  |                     |
|------------------|---------------------|
| output impedance | 75 ohms             |
| output level     | 1 V p-p terminated  |
| sample rate      | 5 MHz               |
| number of bits   | 8                   |
| filter cutoff    | 2 MHz, 12 db/octave |

### 3.4 Composite Sync.

Composite sync is used to drive the video camera. The DVDS uses the EIA RS-170 standard.

|                  |                    |
|------------------|--------------------|
| output impedance | 75 ohms            |
| level            | 4 V p-p terminated |
| polarity         | negative going     |
| horizontal rate  | 15,734 Hz          |
| horizontal lines | 525                |
| active lines     | 480                |
| vertical rate    | 60 Hz              |
| frame rate       | 30/second          |

Note that because this is a 256 H by 240 V system, the same image is repeated on both fields.

### 3.5 Camera requirements

The most important requirement is that the camera sync to the system. A camera that will lock to the above RS-170 format will work. To obtain maximum use of the 256 gray levels, it should have a signal to noise ratio greater than 48 dB. The digitizing rate is 5 MHz so the camera should have a bandwidth of at least 2.5 MHz.

### 3.6 Adjustment procedure

- 1) Needed: small screw driver, dual trace oscilloscope, Drawing # 001-018 A/D-D/A component layout.
- 2) Remove the cover from the machine.
- 3) Connect the camera.
- 4) Turn the power on.
- 5) Point the camera on a bright scene and open the f-stop for maximum contrast.

#### A/D Adjustment

- 6) Set both scope channels for 1V/division.
- 7) Ground both probes and line the traces on the scope screen up on the second line from the bottom. This is 0V.
- 8) Referring to drawing # 001-018, connect Ch 1 to TP-1 and Ch 2 to TP-2. Unground them and sync to Ch 1. Ch 2 should be between 6.5V and 7.0V.
- 9) Cap the camera lens. Turn the BLACK LEVEL ADJUST pot so that the lowest portion of the active video ( not sync ) just touches 0V.

- 10) Uncap the camera lens and turn the FULL SCALE ADJUST pot so that the highest ( brightest ) portions of the active video just touch the line on Ch-2. If the video goes above the CH-2 reference, the white clip light on the front panel should turn on.
- 11) The adjustments in steps 9 and 10 are somewhat interactive so go back and repeat 9 and 10 as required.

#### D/A Adjustment

- 12) Either load a full scale ramp into the reference memory or point the camera at a bright light source to create a full scale signal for the D/A.
- 13) Set CH1 for .5V/division
- 14) Ground the probe and put the trace on the second line up from the bottom.
- 15) Connect the probe to TP-4. Unground the scope an A.C. couple it.
- 16) Turn the D/A REFERENCE ADJUST pot so that the video signal is 1 Volt peak to peak (2 Volts peak to peak if nothing is connected to the output. i.e. unterminated).

This completes the Adjustment procedure.

## 4.0 Mapping RAM

### 4.1 Description

The 256 x 8 Mapping RAM can be loaded with an arbitrary transfer function to process the image stored in the Bit Plane Memory. The output of this RAM can be sent to the analog input/output card for display or to the DMA interface for transmission to the computer. Note that due to the pipeline delay of the circuitry, the displayed image through the Mapping RAM is shifted to the right by one pixel.

#### 4.2 Loading the Mapping RAMs

The following is the general algorithm for entering a transfer function. A dedicated address generator is included for loading. The circuitry has an autoincrement feature so it is not necessary to specify each address when loading successive locations.

1. Set register 224 to 1 - this puts the RAMs into the write mode.
2. Load register 227 with the starting address. This is usually 0.
3. To write into the mapping RAM, load register 225 with the desired data. Loading this register initiates the write function into the RAM.
4. The address will automatically increment, so for successive addresses go to step 3. To load random locations go to step 2.
5. When completed, load register 224 with 0 to put the RAMs into the normal mode.

#### 5.0 Address Generator

##### 5.1 Description

This card generates the video read and write addresses which are then used by the image memories. It also generates the refresh signals for the dynamic RAMs used in the image memory. The addressing modes are fixed.

## 6.0 Computer Interface Connector list

|    |              |             |      |    |                          |      |
|----|--------------|-------------|------|----|--------------------------|------|
| 1  | Data Port    | Bit 0 (LSB) | [1]  | 20 | Computer Ready           | [18] |
| 2  | Data Port    | Bit 1       | [2]  | 21 | Device Busy (Strobe)     | [19] |
| 3  | Data Port    | Bit 2       | [3]  | 22 | GND                      | [20] |
| 4  | Data Port    | Bit 3       | [4]  | 23 | Device Busy (Strobe)Inv. | [21] |
| 5  | Data Port    | Bit 4       | [5]  | 24 | Output Port Bit 0 (LSB)  | [22] |
| 6  | Data Port    | Bit 5       | [6]  | 25 | Output Port Bit 1        | [23] |
| 7  | Data Port    | Bit 6       | [7]  | 26 | Output Port Bit 2        | [24] |
| 8  | Data Port    | Bit 7       | [8]  | 27 | Output Port Bit 3        | [25] |
| 9  | GND          |             | [9]  | 28 | Output Port Bit 4        | [26] |
| 10 | Address Port | Bit 0 (LSB) | [10] | 29 | Output Port Bit 5        | [27] |
| 11 | Address Port | Bit 1       | [11] | 30 | Output Port Bit 6        | [28] |
| 12 | Address Port | Bit 2       | [12] | 31 | Output Port Bit 7        | [29] |
| 13 | Address Port | Bit 3       | [13] | 32 | Device Ready             | [30] |
| 14 | Address Port | Bit 4       | [14] | 33 | Not Connected            | [31] |
| 15 | Address Port | Bit 5       | [15] | 34 | Device Ready (Inv.)      | [32] |
| 16 | Address Port | Bit 6       | [16] | 35 | Device Busy              | [33] |
| 17 | Address Port | Bit 7       | [17] | 36 | Device Busy (Inv.)       | [34] |
| 18 |              |             |      | 37 |                          |      |
| 19 |              |             |      |    |                          |      |

Notes: All signals are TTL levels active high except those marked (Inv.) which are active low. Numbering is for the 37 pin D type subminiature connector on the rear panel. Numbers in brackets are for P1, the 34 pin cable socket on interface board #10001 (drawings 001-002, 001-003, 001-004 001-005 and 001-019).







# DIGITAL/ANALOG DESIGN

ELECTRONIC DESIGN - PRODUCT DEVELOPMENT

---

530 BROADWAY  
NEW YORK, N.Y. 10012  
(212) 966-0410

*Jim H. ...*

DIGITAL VIDEO DEVELOPMENT SYSTEM (DVDS) VERSION 7

OPERATORS MANUAL

FOR: NATIONAL BUREAU OF STANDARDS

PROGRAMMABLE AUTOMATION DIVISION

11/15/82

## TABLE OF CONTENTS

### 1.0 System Introduction

- 1.1 Controller
- 1.2 Image Memories
  - 1.2.1 Reference Memory
  - 1.2.2 Bit Plane Memory
- 1.3 Address Generator
- 1.4 Analog Input/Output
- 1.5 Arithmetic Unit
- 1.6 Computer Interface
- 1.7 Case and Electrical

### 2.0 Computer Interface

- 2.1 Description
- 2.2 List of Registers, Addresses, and Modes
  - 2.2.1 Computer Interface
  - 2.2.2 Arithmetic Unit
  - 2.2.3 Operating Modes
- 2.3 Direct Memory Access (DMA)
  - 2.3.1 Introduction
  - 2.3.2 DMA Read
  - 2.3.3 DMA Write (reference memory only)
- 2.4 Interfacing Requirements for the Computer
  - 2.4.1 Computer Output Ports
  - 2.4.2 Computer Input Port

### 3.0 Analog Input/Output

- 3.1 Description
- 3.2 Analog to Digital Section
- 3.3 Digital to Analog Section
- 3.4 Composite Sync
- 3.5 Camera Requirements
- 3.6 Adjustment Proceedure

### 4.0 Arithmetic Unit

- 4.1 Description
  - 4.1.1 Functions
- 4.2 Loading the Mapping RAMs

### 5.0 Address Generator

- 5.1 Description

### 6.0 Connector List

- 6.1 Computer Interface
- 6.2 Arithmetic Unit Connector
- 6.3 Address Connector

## 7.0 Apple Interface

- 7.1 Introduction
- 7.2 Output Ports
- 7.3 Input Ports
- 7.4 Handshaking Lines

## 1.0 System Introduction

### 1.1 Controller

The controller card contains the master clock oscillator, PROMs with all the necessary timing and sync signals, and the circuitry needed for controlling the dynamic RAMs used in the image memories.

### 1.2 Image Memories

There are two memory cards, the reference and the bit plane. Each memory card uses 16K dynamic RAMs arranged 256 H by 240 V by 8 deep. They run in real time, synchronous with the video display. Since the RAMs do not access fast enough, they are multiplexed 4 deep. For a read cycle, four pixels are addressed in parallel and are serially shifted out. The write cycle works similarly but in the opposite direction.

#### 1.2.1 Reference Memory

The reference memory stores an eight bit gray scale image. It may be filled with data from the video input or directly written into through the DMA interface from the host computer. It may be displayed through the video output or unloaded through the DMA interface to the computer.

#### 1.2.2 Bit Plane Memory

The bit plane memory is configured as eight 256 X 240 planes. Input data to any of the planes comes from a magnitude comparator. This compares the live video input with the output of the reference memory on a pixel by pixel basis. If the input pixel is less than or equal to the reference pixel, a zero is stored. If it is greater, a one is stored. Only one plane may be written into at a time. This memory may be displayed through the video output or unloaded to the computer through the DMA interface. The arithmetic unit may be placed between the bit plane memory and either of these output channels (see 1.6).

### 1.3 Address Generator

This card generates the addresses used to read from and write to the memory in the video modes. It also generates the refresh addresses for the dynamic RAMs. Its addressing modes are fixed.

### 1.4 Analog Input/Output

This card is the interface from the camera to the frame store and from the frame store to the CRT monitor. The video input from the camera is amplified and digitized at 5 MHz to 8 bit gray

scale resolution. The digitized video is then put onto the system bus destined for the memory. There is a selector that allows the viewing of any of three sources and either the entire gray scale or any of the eight individual planes. The translation from digital to video is done at 5 MHz in an 8 bit digital to analog converter.

#### 1.5 Arithmetic Unit

The arithmetic unit (AU) is used to process the output of the bit plane memory. It can perform both arithmetic and logic functions. RAMs loaded from the computer are included for mapping both operands.

#### 1.6 Computer Interface

The computer interface card is the link between the frame store and the host computer. It uses an 8 bit bidirectional data bus and an 8 bit address bus. This card allows the user to program the frame store into its various modes. It also has its own address generator for the image memory that allows bidirectional Direct Memory Access (DMA).

#### 1.7 Case and Electrical

The Frame store is packaged in an S-100 type computer case. The motherboard has been modified by adding terminating resistors. It uses standard S-100 wire wrap cards. There is an internal power supply that uses 120 VAC  $\pm$  10%. The on/off switch has a key lock and is located, along with the indicator light, on the front panel. Due to several internal reset circuits, after turning the unit off, the user should wait 10 seconds before turning the unit back on. All video connections are made with BNC connectors located on the back panel. The computer interface is via a 37 pin subminiature D type connector on the back panel. The fuseholder is also on the back panel. The fuse is a 3 amp slo-blo type. Additionally, the three unregulated voltages on the bus are individually fused. The +8 volt line uses a 7 amp fuse while the +18 and -18 lines use 2 amp fuses.

### 2.0 Computer Interface

#### 2.1 Description

The DVDS is designed to be a peripheral to a host computer. Its modes of operation are programmable and once programmed it needs no further intervention to operate. Programming is done by loading registers in the DVDS that control specific functions.

## 2.2 List of Registers, Addresses, and Modes

(Register)  
Address

```

1      Bit 0 : 0 = Reference Memory Live
           1 = Reference Memory Frozen
      Bit 1 : 0 = Bit Plane Memory Frozen
           1 = Bit Plane Memory Live
      Bit 2 : 0 = DMA Output Mode
           1 = DMA Input Mode
      Bit 3 : 0 = Normal (Video IN, OUT) Mode
           1 = DMA Mode
      All other bits are "don't care"

```

### 3 DMA Vertical Address Register (0-255)

5      Bit 0 : 0 = Display Reference Memory  
              1 = Display Bit Plane Memory  
            Bit 1 : 0 = Direct Memory Display  
                    1 = Display Bit Plane Memory through Arithmetic  
                      Unit (A.U.)

6

Bit 4 : 0 = A/D on Input Data Bus  
 1 = Command Data Bus on Input Data Bus  
 Note: This allows Data from the Host Computer to be placed on the Bit Plane Memory Input rather than the Input Video. This is useful for testing.

All other Bits are "don't care"

6 Bits 0-2 : Writing into the Bit Plane Memory occurs one plane at a time. These three bits determine which plane (0-7) is active (live). For the display of a single bit of one of the output sources (Register 5 Bit 3 High) register 6 selects the bit to be displayed.

All other Bits are "don't care"

## 2.2.2 Arithmetic Unit

### Address

224 Bit 0 : 0 = Normal Operation  
 1 = Load Masking RAMs  
 Bits 1-4: "Don't care"  
 Bits 5-7: Arithmetic Function  
 000 = CLEAR  
 001 = Horizontal Line Number Minus Bit Plane  
 Memory  
 010 = Bit Plane Memory Minus Horizontal Number  
 011 = Bit Plane Memory Plus Horizontal Line  
 100 = Bit Plane Memory EXOR Horizontal Line  
 101 = Bit Plane Memory OR Horizontal Line  
 110 = Bit Plane Memory AND Horizontal Line  
 111 = PRESET

225 write enable for Image Mask RAMs (0-255)

226 write enable for Line Mask RAMs (0-255)

227 write address register (0-255)

255 *Positive Factor* (0-255)

## 2.2.3 Operating Modes

|    | Mode  | Register 1 | Register 5 |
|----|---|------------|------------|
| 1. | Both Memories Frozen,<br>Reference Memory Displayed.                              | 1          | 0          |
| 2. | Reference Memory Live, Bit<br>Plane Memory Frozen, Reference<br>Memory Displayed. | 0          | 0          |



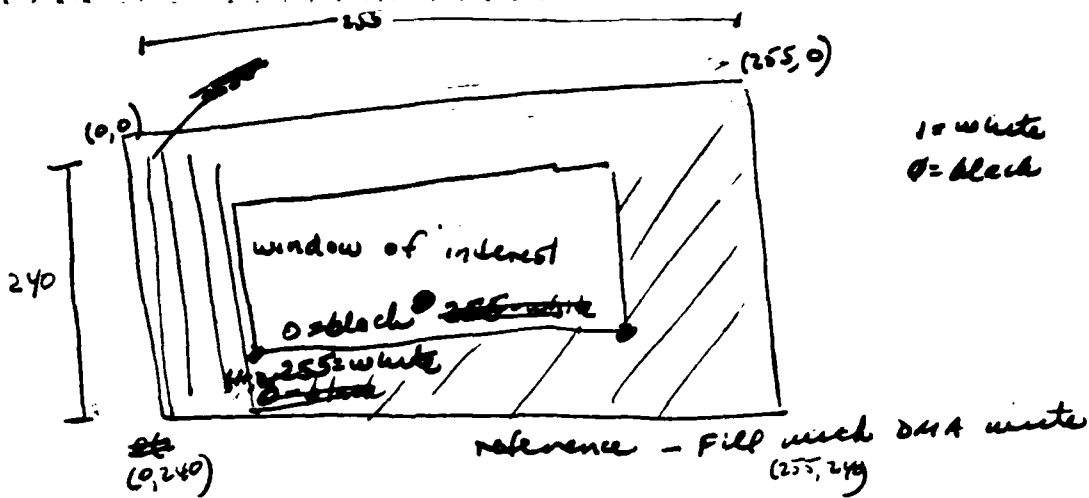
|   |    |   |
|---|----|---|
| 3. Both Memories Frozen,<br>Bit Plane Memory Displayed.   | 1  | 1 |
| 4. Bit Plane Memory Live, Reference<br>Memory Frozen, Bit Plane Memory<br>Displayed, Register 6 contains the<br>Plane Number which is Live (0-7).   | 3  | 1 |
| 5. Both Memories Frozen,<br>Display Bit Plane Memory through<br>Arithmetic Unit.  | 1  | 3 |
| 6. Bit Plane Memory Live,<br>Display Bit Plane Memory through<br>Arithmetic Unit. Register 6<br>contains the Plane Number which<br>is Live (0-7). Register 224<br>contains A.U. function. | 3  | 3 |
| 7. Both Memories Frozen, Reference<br>Memory Displayed, DMA Write into<br>Reference Memory.   | 8  | 0 |
| 8. Both Memories Frozen, Reference<br>Memory Displayed, DMA read from<br>Reference Memory.  | 12 | 0 |
| 9. Both Memories Frozen, <del>Reference</del><br>Memory Displayed, DMA read from<br>Bit Plane Memory.   | 12 | 1 |
| 10. Both Memories Frozen, Reference<br>Memory Displayed, DMA read from<br>Bit Plane Memory through the<br>Arithmetic Unit. Register 224<br>contains A.U. function.                        | 12 | 3 |

Note: In any of these modes a single plane of the output source may be viewed by adding 4 to the number in Register 5 and loading Register 6 with the Plane number (0-7)

### 2.3 Direct Memory Access (DMA)

#### 2.3.1 Introduction

The DVDS allows the host computer to directly access the memories. Frozen images from either memory or from the bit plane memory through the A.U. may be unloaded to the computer via the high speed DMA interface. Additionally, the reference memory can be directly loaded from the computer. The transfer rate in either direction can exceed 200K bytes/second. A DMA address



select bit plane

freeze

bit plane should have windowed view

Inputs

leftwin - vertical

rightwin - vertical

topwin - horiz. [max=255]

botwin - horiz. [min=0]

horizontal address

0 → 2

next address

0 → 3

DMA write

8 → 1

max 64, 240

max 64, 255

generator is used for both read and write operations. Pixel coordinates may be loaded into the horizontal and vertical address registers for random access of the image memory. Additionally, the DMA address generator will autoincrement after each read or write operation. This allows a starting address to be specified and then the circuit will provide sequential addresses itself. This speeds the data transfer rate substantially. The top left pixel is at address 0 H, 0 V. The top right pixel is at address 255 H, 0 V. The bottom left pixel is at address 0 H, 240 V. The bottom right pixel is at location 255 H, 240 V. The addresses will autoincrement from the last address loaded into the horizontal address register until 255 is reached. Then the vertical address will increment by one and the horizontal address overflows back to 0.

### 2.3.2 DMA Read

The algorithm for performing DMA read is outlined below. See also the program listings for REF.RAMP and MOVLOD.OBJO

1. load register 0 with 1 - allows DMA access during entire frame.
2. load register 1 with  $1\frac{3}{4}$  - this freezes the images and prepares for a DMA read operation.
3. load register 5 with the proper code for the desired output source (0 = Reference Memory, 1 = Bit Plane Memory, 3 = Bit Plane Memory through the ALU)
4. load register 2 (DMA horizontal address register) with H pixel coordinate.
5. load register 3 (DMA vertical address register) with V pixel coordinate.
6. load register 4 to assert the DMA read at the previously specified address. The pixel will be sent to the computer via the DVDS output port.
7. for random access go to 4. for sequential access go to 6. and repeat until the desired number of pixels are recieved.
8. load register 1 with 1. - this returns the system to its normal state and keeps the images frozen.

Note: The handshaking lines should be checked after each step

### 2.3.3 DMA Write (Reference Memory Only)

The procedure for performing DMA write is similar to the DMA read. Also see the program listing REF.READ and MOVLOD.OBJO

1. load register 0 with 1 - this allows DMA access during the entire frame.
2. load register 1 with 8. - this freezes the image and prepares the unit for write.
3. Load register 5 with 0 - this displays the reference memory
4. load register 2 with the pixel H coordinate.
5. load register 3 with the pixel V coordinate.
6. load register 4 with the pixel to be written into the image memory.
7. for random access go to 4. for sequential access go to 6. and repeat until the desired number of pixels are transferred.
8. load register 1 with 1 - this returns the unit to normal operation with the images frozen.

Note: The handshaking line should be checked after each step

## 2.4 Interfacing Requirements for the Computer

### 2.4.1 Computer Output Ports

Two output ports must be provided by the computer, the command port and the data port. Both must be 8 bits wide, active high at normal TTL levels. All lines are buffered in the DVDS and present 1<sup>LS</sup> TTL load. 470 ohm pull up resistors to +5 volts are included on these lines in the DVDS. These ports are treated by the unit as one 16 bit port. If separate ports are used it is important to load the data port before the command port, which has the handshaking lines associated with it. Once data has been set up and latched on both ports the COMPUTER READY line should go high. The DVDS will latch both ports soon after the rising edge of this signal. COMPUTER READY should be a standard TTL level signal. At this time the DEVICE BUSY line will go High to indicate that the command is being executed. The value loaded into register 0 determines when commands will be executed. In the "V.Drive Only" mode (0), a command will be held until the next

vertical blanking interval before execution. This is useful for freezing the memories at the end of a field. Other commands such as DMA transfers should take place at high speed so register 0 is put into the "Anytime" mode (1). Once the DVDS has executed the command from the host computer, the DEVICE BUSY line (TTL) will go low. Also at this time the BUSY strobe will pulse for 1 microsecond. These indicate that the computer may send another command. Any commands sent before this time will be lost.

#### 2.4.2 Computer Input Port

One 8 bit input port to the computer is required to read data from the image memory. This port must be standard level, TTL, active high. After a DMA READ command has been executed, the pixel datum will be placed on the computer input port. The DEVICE READY line will then pulse for 1 microsecond to strobe this information into the computer input port. This signal is at standard TTL levels. Note that the data should be strobed in on the trailing edge of the DEVICE READY line.

### 3.0 Analog Input/Output

#### 3.1 Description

This card accepts the video input from a camera, digitizes it, and passes it to the image memory. It also receives digital video from various sources, selects the source, converts it to analog video, and passes it to the CRT monitor. In addition, composite sync is available for driving the camera. BNC connectors for video in, video out, and composite sync are located on the back of the DVDS.

#### 3.2 Analog to Digital section

This section accepts the input video from the camera and digitizes it.

|                       |                          |
|-----------------------|--------------------------|
| input impedance       | 75 ohms                  |
| input signal level    | 1 volt sync tip to white |
| sample rate           | 5 MHz                    |
| gray level resolution | 256 levels               |
| number of bits        | 8                        |

A white clip light is provided on the front panel. This will come on when the video signal is too high, causing the analog to digital converter to overflow.

### 3.3 Digital to Analog section

Either of the stored images or the arithmetic unit may be selected for display. In addition to the full eight bit grey scale display, a single bit plane of any of these sources may be selected for viewing with the bit plane address register (see 2.2.1). The single bit mode output is either full black or full white. Blanking and sync are added digitally so there is no adjustment of sync or set up levels.

|                  |                    |
|------------------|--------------------|
| output impedance | 75 ohms            |
| output level     | 1 V p-p terminated |
| sample rate      | 5 MHz              |
| number of bits   | 8                  |

### 3.4 Composite Sync

Composite sync is used to drive the video camera. The DVDS uses the EIA RS-170 standard.

|                  |                    |
|------------------|--------------------|
| output impedance | 75 ohms            |
| level            | 4 V p-p terminated |
| polarity         | negative going     |
| horizontal rate  | 15,734 Hz          |
| horizontal lines | 525                |
| active lines     | 480                |
| vertical rate    | 60 Hz              |
| frame rate       | 30/second          |

Note that because this is a 256 H by 240 V system, the same image is repeated on both fields.

### 3.5 Camera requirements

The most important requirement is that the camera sync to the system. A camera that will lock to the above RS-170 format will work. To obtain maximum use of the 256 gray levels, it should have a signal to noise ratio greater than 48 dB. The digitizing rate is 5 MHz so the camera should have a bandwidth of at least 2.5 MHz.

### 3.6 Adjustment procedure

- 1) Needed: small screw driver, dual trace oscilloscope, Drawing #F-026 A/D-D/A component layout.
- 2) Remove the cover from the machine.
- 3) Connect the camera.
- 4) Turn the power on.
- 5) Point the camera on a bright scene and open the f-stop for maximum contrast.



#### A/D Adjustment

- 6) Set both scope channels for 1V/division.
- 7) Ground both probes and line the traces up on the second line from the bottom. This is 0V.
- 8) Referring to drawing # P-026, connect Ch 1 to TP-1 and Ch 2 to TP-2. Unground them and sync to Ch 1. Ch 2 should be at approximately +6.7V.
- 9) Cap the camera lens. Turn the CLAMP LEVEL ADJUST pot so that the lowest portion of the active video ( no sync ) just touches 0V.
- 10) Uncap the camera lens and turn the FULL SCALE ADJUST pot so that the highest ( brightest ) portions of the active video just touch the 6.7V line on Ch-2. If the video goes above the CH-2 reference, the white clip light on the front panel should turn on.
- 11) The adjustments in steps 9 and 10 are somewhat interactive so go back and repeat 9 and 10 as required.

#### D/A Adjustment

- 12) Set CH1 for .5V/division
- 13) Ground the probe and put the trace on the scope centerline.
- 14) Connect the probe to TP-3. Unground the scope.
- 15) Turn the D/A REFERENCE ADJUST pot so the scope reads - 1.0 Volts.
- 16) Connect the probe to TP-4. AC couple the scope.
- 17) Turn the output LEVEL ADJUST pot so that the video signal is 1 Volt peak to peak (2 Volts peak to peak if nothing is connected to the output. i.e. unterminated).

This completes the Adjustment procedure.

## 4.0 Arithmetic Unit

### 4.1 Description

The heart of the unit is eight bit ALU (2 X 74S381). The two operands to the ALU are an output pixel from the bit plane memory and the vertical line (address) of that pixel. The pixel is first passed through a PROM that is programmed to perform a gray code to binary conversion. It is then passed through a 256 X 8 mapping RAM to perform an arbitrary transfer function. The other operand, the line number, is also passed through an identical mapping RAM before going to the ALU. The output of the arithmetic unit can be sent to the analog input/output card for display or to the DMA interface and then to the computer. Note that due to the pipeline delay of the processor, the displayed image through the AU is shifted to the right by one pixel.



#### 4.1.1 Functions

The ALU can perform the following functions:

- 1) clear
- 2) line - memory
- 3) memory - line
- 4) memory + line
- 5) memory EXOR line
- 6) memory OR line
- 7) memory AND line
- 8) preset

Note that no provision for numeric overflow or underflow is made. See section 2.2.2 for detailed programming information.

#### 4.2 Loading the Mapping RAMs

*Set  $\Phi$  = high order in mask*

The following is the general algorithm for entering a transfer function. The RAMs use a common address generator for loading. The circuitry has an autoincrement feature so it is not necessary to specify each address when loading successive locations.

1. Set register 224 to 8 - this puts the RAMs into the write mode. *1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32*
2. Load register 227 with the starting address. This is usually 0.
3. To write into the image mapping RAM, load register 225 with the desired data. To write into the line mapping RAM, load register 226 with the desired data. Loading these registers performs the write function in the RAM.
4. The address will automatically increment, so for successive addresses go to step 3. To load random locations go to step 2.
5. When completed, load register 224 with a number representing the desired ALU function.

#### 5.0 Address Generator

##### 5.1 Description

This card generates the video read and write addresses which are then used by the image memories. It also generates the refresh signals for the dynamic RAMs used in the image memory. The addressing modes are fixed.

## 6.0 Connector list

### 6.1 Computer Interface - P1 on L(N)-011 to P1 on F-035

|    |              |             |      |    |                          |      |
|----|--------------|-------------|------|----|--------------------------|------|
| 1  | Data Port    | Bit 0 (LSB) | [1]  | 20 | Computer Ready           | [18] |
| 2  | Data Port    | Bit 1       | [2]  | 21 | Device Busy (Strobe)     | [19] |
| 3  | Data Port    | Bit 2       | [3]  | 22 | GND                      | [20] |
| 4  | Data Port    | Bit 3       | [4]  | 23 | Device Busy (Strobe)Inv. | [21] |
| 5  | Data Port    | Bit 4       | [5]  | 24 | Output Port Bit 0 (LSB)  | [22] |
| 6  | Data Port    | Bit 5       | [6]  | 25 | Output Port Bit 1        | [23] |
| 7  | Data Port    | Bit 6       | [7]  | 26 | Output Port Bit 2        | [24] |
| 8  | Data Port    | Bit 7       | [8]  | 27 | Output Port Bit 3        | [25] |
| 9  | GND          |             | [9]  | 28 | Output Port Bit 4        | [26] |
| 10 | Address Port | Bit 0 (LSB) | [10] | 29 | Output Port Bit 5        | [27] |
| 11 | Address Port | Bit 1       | [11] | 30 | Output Port Bit 6        | [28] |
| 12 | Address Port | Bit 2       | [12] | 31 | Output Port Bit 7        | [29] |
| 13 | Address Port | Bit 3       | [13] | 32 | Device Ready             | [30] |
| 14 | Address Port | Bit 4       | [14] | 33 |                          | [31] |
| 15 | Address Port | Bit 5       | [15] | 34 | Device Ready (Inv.)      | [32] |
| 16 | Address Port | Bit 6       | [16] | 35 | Device Busy              | [33] |
| 17 | Address Port | Bit 7       | [17] | 36 | Device Busy (Inv.)       | [34] |
| 18 |              |             |      | 37 |                          |      |
| 19 |              |             |      |    |                          |      |

Notes: All signals are TTL levels active high except those marked (Inv.) which are active low. Numbering is for the 37 pin D type subminiature connector on the rear panel. Numbers in brackets are for the 34 pin cable socket on L(N)-011 and F-035

## 6.2 Arithmetic Unit Connector - P2 on L(N)-011 to P1 on N-028

|    |                              |       |                          |
|----|------------------------------|-------|--------------------------|
| 1  | DMA Vertical Address 0 (LSB) | 11    | ALU Output Port Bit 2    |
| 2  | DMA Vertical Address 0       | 12    | ALU Output Port Bit 3    |
| 3  | DMA Vertical Address 2       | 13    | ALU Output Port Bit 4    |
| 4  | DMA Vertical Address 3       | 14    | ALU Output Port Bit 5    |
| 5  | DMA Vertical Address 4       | 15    | ALU Output Port Bit 6    |
| 6  | DMA Vertical Address 5       | 16    | ALU Output Port Bit 7    |
| 7  | DMA Vertical Address 6       | 17    | DMA ALU LATCH ENABLE     |
| 8  | DMA Vertical Address 7       | 18    | ALU LATCH CLOCK          |
| 9  | ALU Output Port Bit 0 (LSB)  | 19    | Output Port Select (OPS) |
| 10 | ALU Output Port Bit 1        | 20-34 | Spare                    |

## 6.3 Address Connector - P2 on N-028 to P1 on L-007

|   |                               |      |                         |
|---|-------------------------------|------|-------------------------|
| 1 | Read Vertical Address 0 (LSB) | 6    | Read Vertical Address 5 |
| 2 | Read Vertical Address 1       | 7    | Read Vertical Address 6 |
| 3 | Read Vertical Address 2       | 8    | Read Vertical Address 7 |
| 4 | Read Vertical Address 3       | 9-16 | Spare                   |
| 5 | Read Vertical Address 4       |      |                         |

## 7.0 Apple interface

### 7.1 Introduction

The Apple Interface contains the hardware necessary to interface the DVDS to an Apple II microcomputer. It provides for two 8 bit output ports, one 8 bit input port and all necessary handshaking lines. It normally resides in slot #4 of the Apple's peripheral bus.

### 7.2 Output Ports

The output ports are memory mapped locations. The first port contains the address of the DVDS register to be loaded. This is called the Command Address Port. When the card is in slot #4 its

address is 49356 (COCC HEX). The second port sends the data to be loaded into the addressed register. This is called the Command Data Port. With the card in slot #4 its address is 49357 (COCD Hex).

### 7.3 Input Ports

The input port is used to receive DMA data from the DVDS. In slot #4 its address is 49344 (COC1 HEX). Data is strobed into this memory mapped register by the DEVICE READY (Inv.) line from the DVDS. It then can be read by the computer.

### 7.4 Handshaking Lines

A single COMPUTER READY line is provided for both output ports. The decoded write pulse for the Command Address Port also serves as the COMPUTER READY line. Thus the Command Data Port is loaded first followed by the Command Address Port. Loading this register signals the DVDS that a command must be executed.

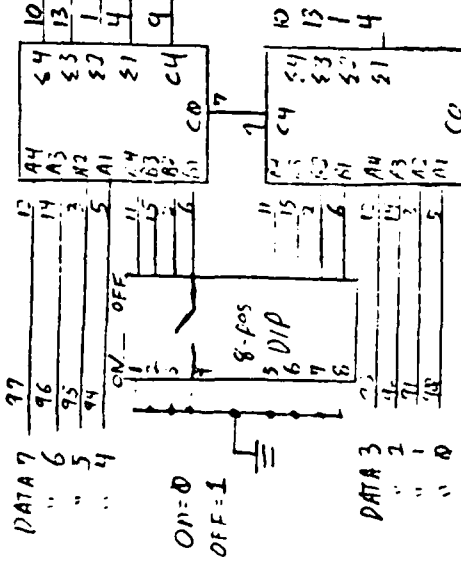
The Apple Interface uses two handshaking lines provided by the DVDS, DEVICE BUSY and DEVICE READY STrobe (Inv.). Both are mapped into memory location 49345 (COC1 HEX), which is the status register. DEVICE BUSY goes active (high) upon activation of the COMPUTER READY line. It remains high until the DVDS has completed execution of the command. It then goes low. DEVICE BUSY is bit 1 of the status register.

When the DVDS performs a DMA Read command it leaves the data in the input register and sets (forces high) a flag connected to bit 0 of the status register. The Apple may monitor this bit to wait until the DMA transfer is complete. Reading the input register resets this bit to ready it for the next transfer.



OUTPUT DATA

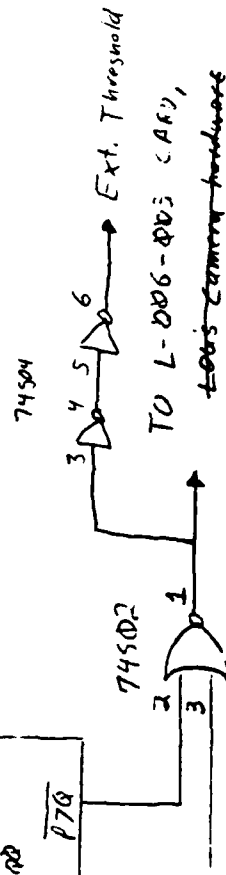
745283



74LS684

INPUT DATA

|    |    |        |
|----|----|--------|
| 17 | 47 | DATA 7 |
| 16 | 46 | " 6    |
| 15 | 45 | " 5    |
| 14 | 44 | " 4    |
| 13 | 43 | " 3    |
| 12 | 42 | " 2    |
| 11 | 41 | " 1    |
| 10 | 40 | " 0    |



= 1 whenever input is positive  
is greater than sum of reference  
memory and DIP setting

Blue Box only

DAD box wiring

1V29/83

| <u>Signals</u>    | <u>output pins</u>                     |
|-------------------|--|
| INTRB             | N.C.                                   |
| OBF <sub>B</sub>  | 18                                     |
| ACK <sub>B</sub>  | 30, 32                                 |
| INTR <sub>A</sub> | N.C.                                   |
| BIT 4             | N.C.                                   |
| BIT 5             | N.C. (earlier versions → 24 for flash) |
| ACK <sub>A</sub>  | 30, 32                                 |
| OBF <sub>A</sub>  | N.C.                                   |
| INTR <sub>B</sub> | N.C.                                   |
| IBF <sub>B</sub>  | N.C.                                   |
| STB <sub>B</sub>  | N.C.                                   |
| INT 2             | N.C.                                   |
| STB <sub>A</sub>  | 26                                     |
| IBF <sub>A</sub>  | 20                                     |
| BIT 6             | N.C.                                   |
| BIT 7             | 32 (for VSYNC if required)             |

use approx. wiring on 8255A2 (use 7426 on  
X's output iff you need to drive the flash from  
bit 5, output)

# ISBX 350 wiring for output to OAD box

11/24/83

## Function

E1-E2 port A output

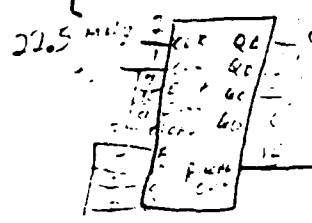
|        |                      |                         |
|--------|----------------------|-------------------------|
| INTRB  | E7- <del>E8</del> NC | N.C.                    |
| OBFB   | E9-E14               | Computer Ready (pin 18) |
| ACKB   | E11-E26              | <u>Device Busy</u>      |
| INTR A | E13- <del>E</del> NC | N.C.                    |
| BIT 4  | E19-E20              | N.C.                    |
| BIT 5  | E21-E22              | N.C.                    |
| ACKA   | E23- <del>E</del> 24 | <u>Device Busy</u>      |
| OBFA   | E25-NC               | N.C.                    |
|        | E24-E26              | connect ACKA, ACKB      |



74LS14 10/13/83 76w

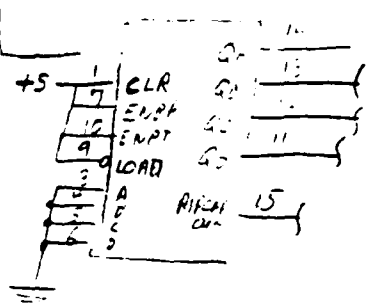
74S162 0.01 uf

5T= 5 16 2 comp  
1-17 1 14



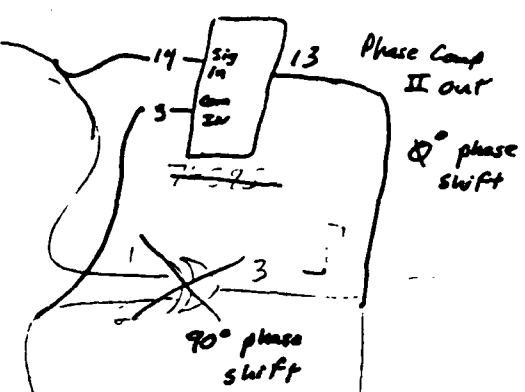
2.25 MHz

74S162

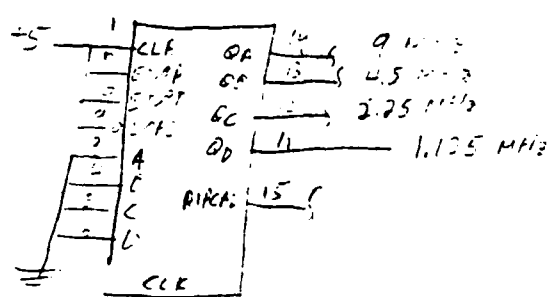


1.125 MHz

CD4046



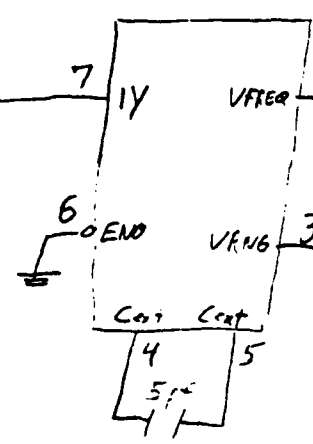
74S162



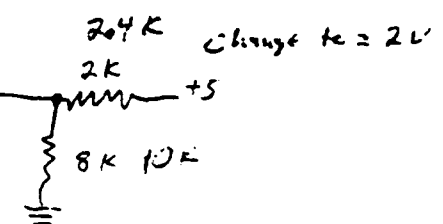
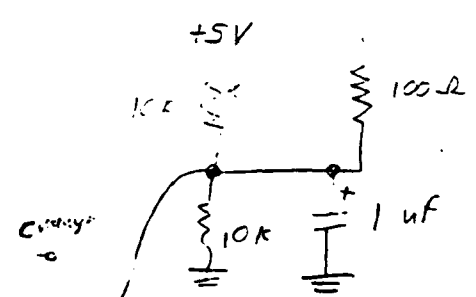
74LS124

Disconnected  
10/13/83 18 MHz  
TEN

18.018 MHz



16, 15 +5V  
8, 7 GND



Change to = 2V

APPENDIX D

SOFTWARE FOR FRANKLIN

## LIST

```
1 DIM RANGEZ(1,256)
2 LINE = 328
3 MASH = 325
4 AREG = 227
5 ONERR GOTO 10
6 GOSUB 1000: REM ...INIT
10 HOME
20 PRINT "WELCOME TO THE DAKESIDE...."
30 VTAB 10
35 PRINT "0 - END
40 PRINT "1 - LOAD IMAGES
50 PRINT "2 - DISPLAY RANGES
55 PRINT "3 - LOAD LUTS
56 PRINT "4 - LOAD FLASH SYNC
57 PRINT "5 - LOAD RANGE TABLE
58 PRINT "6 - INIT SYSTEM
59 PRINT "7 - SET THRESHOLD
60 PRINT "8 - DISPLAY BP (16)
61 PRINT "9 - DISPLAY BP ALU
90 VTAB 20
100 GET IN#
110 IF ASC (IN#) = 49 THEN GOSUB 1700
120 IF ASC (IN#) = 50 THEN GOSUB 40000
130 IF ASC (IN#) = 51 THEN GOSUB 10000
140 IF ASC (IN#) = 52 THEN GOSUB 20000
150 IF ASC (IN#) = 53 THEN GOSUB 50000
157 FOR D,10
160 IF ASC (IN#) = 48 THEN GOTO 63000
170 IF ASC (IN#) = 54 THEN GOSUB 1000
180 IF ASC (IN#) = 55 THEN GOSUB 30000
190 IF ASC (IN#) = 56 THEN GOSUB 25000
200 IF ASC (IN#) = 57 THEN GOSUB 26000
500 GOTO 10
1000 REM ...THE SYSTEM WILL BE
1020 REM ...KNOWN STATE. REF MEM
1030 REM ...WILL THEN BE LOADED
1040 REM ...WITH A CURRENT IMAGE
1050 REM ...AND SUBSEQUENT BIT
1060 REM ...PLAIN IMAGES ALSO
1070 REM ...LOADED. THIS MAY
1080 REM ...WITH OR WITHOUT THE
1090 REM ...FLASH LOOKUP TABLES
1100 REM ...INITIALIZED TO A
1101 REM ...WILL ALSO BE LOADED
1140 REM ...WILL THEN DISPLAY
1150 REM ...A SMALL WINDOW REF-
1160 REM ...PRESENTING SAMPLED
1170 REM ...AREA IN TERMS OF
1180 REM ...DISTANCE.
1190 REM
1200 REM
1210 REM
1220 REM
1230 REM
1240 REM
```

```

1250 REM ...SYSTEM INITIALIZATION
1260 REM
1270 REM
1280 D = 49357
1290 A = 49356
1300 REM ...D IS THE DATA PORT.
1310 REM ...A IS THE COM PORT.
1320 REM
1330 REM
1340 POKE D,1: POKE A,0
1341 REM
1350 REM ...SET TO "LOAD ANY-
1360 REM ...TIME" DURING V-
1370 REM ...TRACE.
1380 REM
1381 FOR I = 1 TO 20: NEXT
1382 REM
1383 REM ...ALLOW TIME TO
1384 REM ...ACCEPT LOAD COMMAND
1400 REM
1410 THRESH = 0
1420 POKE THRESH,0: POKE A,255
1430 POKE D,0: POKE A,1
1440 POKE D,1: POKE A,2
1450 POKE D,0: POKE A,3
1460 POKE D,0: POKE A,5
1470 POKE D,96: POKE A,224
1480 REM ...SET THRESH TO 0
1490 REM ...SET REF LIVE,BIT
1500 REM ...PLANE FROZEN.
1510 REM ...SET DMA HORIZONTAL
1520 REM ...ADDRESS = 0
1530 REM ...SET DMA VERTICAL
1540 REM ...ADDRESS = 0
1550 REM ...SET DISPLAY TO
1560 REM ...AND GRAYSCALE MODE.
1570 REM ...SET ALU TO ADD MODE.
1580 REM
1590 REM
1600 RETURN
1601 REM
1620 REM
1630 REM
1635 HOME
1640 PRINT "SELECT NORMAL OR "
1650 PRINT "FLASH MODE. "
1660 PRINT "NORMAL = N"
1670 PRINT "FLASH = F "
1675 GET IN$
1676 FLSH = 0
1680 IF ASC (IN$) = 70 THEN FLSH = 1
1690 REM
1700 REM ...LOAD IMAGE.
1720 REM
1730 REM
1740 HOME
1750 PRINT "MANUAL LOAD = M"
1760 PRINT "REALTIME LOAD = R"
1770 GET IN$
1771 REM
1772 POKE D,0: POKE A,5
1780 HOME
1790 PRINT "ALIGN REF MASK"
1800 PRINT ""
1810 PRINT ""
1820 PRINT "PRESS A KEY TO CONTINUE"

```

```
1830 PRINT "LIVE   = TAB"
1840 PRINT "NEXT   = RETURN"
1850 GET IN$
1860 IF ASC (IN$) < > 32 THEN 1920
1870 CALL 20480
1880 REM ...FREEZE REF MEN
1890 GOTO 1780
1900 REM
1910 REM
1920 IF ASC (IN$) < > 9 THEN 1980
1930 POKE D,0: POKE A,1
1940 REM ...REF LIVE (REPEAT)
1950 REM
1960 REM
1970 GOTO 1780
1980 IF ASC (IN$) < > 13 THEN 2000
1990 REM
2000 GOTO 3000: REM ...EXIT
2010 GOTO 1780: REM ...TRAP
2100 REM -----
2200 REM
2300 REM
2400 REM
2500 REM
2600 REM ...LOAD BIT PLANES
2700 HOME
2800 POKE D,5: POKE A,5
2900 POKE D,1: POKE A,1
3000 REM
3100 REM ...BIT PLANE FROZEN
3200 REM ...BIT PLANE DISPLAYED
3300 FLI
3400 FOR I = 0 TO 7
3500 REM
3600 REM ...SELECT PLANE
3700 REM
3800 POKE D,(I): POKE A,6
3900 REM
4000 HOME
4100 PRINT "SPACE = FREEZE
4200 PRINT "TAB   = REPEAT
4300 PRINT "RETURN = NEXT
4400 PRINT "PLANE# =", (I + 1)
4500 REM
4600 REM
4700 GET IN$
4800 REM
4900 REM
5000 REM
5100 REM -----
5200 IF ASC (IN$) < > 32 THEN 5300
5310 CALL 20480
5320 REM ...FREEZE PLANE
5330 REM
5340 GOTO 4200
5350 REM -----
5360 REM
5370 REM
5380 REM -----
5390 IF ASC (IN$) < > 9 THEN 5400
5410 POKE D,3: POKE A,1
5420 REM ...BIT PLANE LIVE
5430 REM
5440 REM
5450 GOTO 4200
5460 REM -----
5470 REM
5480 REM
5490 REM
```

```

5400 REM
5410 IF ASC (IND) > 13 THEN 4200
5420 NEXT I
5440 REM
5450 REM
5500 POKE D,3: POKE A,5
5600 REM ...DISPLAY FROZE REF
6000 REM
6100 RETURN
10000 REM -----
11000 REM
11100 HOME
11105 VTAB 10
11110 PRINT "LOADING L.U. TABLES
12000 REM ...LOAD BASE RAM
13000 REM
14000 REM
15000 POKE D,1: POKE A,224
15100 REM ...SET TO LOAD MODE
15200 POKE D,0: POKE A,227
15300 REM ...INIT RAM ADDRESS
15400 REM
15410 REM ...LOAD LINE RAM
15420 REM
15500 FOR I = 0 TO 127
15600 FOR J = 0 TO 1
15700 POKE D,I
15800 POKE A,LINE
15900 NEXT J
16000 NEXT I
16100 REM
16200 REM
16300 REM
16400 REM ...LOAD BASE RAM
16500 REM
16600 POKE D,0: POKE A,227
16700 FOR I = 127 TO 0 STEP -1
16800 FOR J = 0 TO 1
16900 POKE D,I
17000 POKE A,BASE
17100 NEXT J
17200 NEXT I
17300 REM
17400 REM
17500 POKE D,96: POKE A,224
17600 REM ...SET MODE NORMAL
17700 REM
17800 REM
19999 RETURN
20000 REM -----
20100 REM ...LOADS FLASH CODE
20110 REM
20115 HOME
20120 REM
20125 PRINT "LOADING.....
20130 FOR I = 0 TO 47
20140 READ TEST
20150 POKE (20400 + I),TEST
20155 NEXT I
20160 DATA 175
20170 DATA 50
20180 DATA 192
20190 DATA 30
20200 DATA 5
20210 DATA 25
20220 DATA 0

```

```

20230 DATA 80
20240 DATA 173
20250 DATA 98
20260 DATA 192
20270 DATA 43
20280 DATA 251
20290 DATA 234
20300 DATA 169
20310 DATA 0
20320 DATA 162
20330 DATA 0
20340 DATA 160
20350 DATA 0
20360 DATA 169
20370 DATA 1
20380 DATA 141
20390 DATA 205
20400 DATA 172
20410 DATA 141
20420 DATA 88
20430 DATA 192
20440 DATA 141
20450 DATA 89
20460 DATA 192
20470 DATA 162
20480 DATA 13
20490 DATA 169
20500 DATA 255
20510 DATA 136
20520 DATA 208
20530 DATA 253
20540 DATA 202
20550 DATA 208
20560 DATA 240
20570 DATA 141
20580 DATA 204
20590 DATA 192
20600 DATA 141
20710 DATA 88
20720 DATA 192
20730 DATA 96
20740 REM
20750 REM
20760 RETURN
20770 REM =====
25000 POKE D,3: POKE A,5
25010 RETURN: REM DISPLAY BP (160
25000 POKE D,1: POKE A,5
26100 RETURN
26000 REM *****
26100 REM
26200 REM
26300 REM ...THRESHOLD SETTING
26400 REM
26500 HOME
26600 PRINT "THRESHOLD = ",THRESH
26700 VTAB 10
26800 INPUT "NEW = ";THRESH
26900 POKE D,THRESH: POKE A,255
27000 VTAB 15
27100 PRINT "TAB = EXIT"
27200 GET J11
27300 IF ASC (J11) = 9 THEN 25000
27400 GOTO 26000
28000 RETURN
40000 REM *****

```

```

40100 REM
40200 REM
40300 REM ...DISPLAY RANGES
40400 POKE D,13: POKE A,1
40500 POKE D,3: POKE A,5
40600 REM
40700 HOME
40800 PRINT "(X,Y)      "X,Y
40900 REM
41000 REM ...INA MODE
41100 HOME
41200 HOME
41300 X = (X - 4)
41400 Y = (Y - 10)
41500 FOR J = 0 TO 7
41600 FOR I = 0 TO 19
41700 RTAB (IJ + 1) * 4)
41800 POKE D,(Y + 1)
41900 POKE A,3
42000 POKE D,(X + J)
42100 POKE A,2
42200 POKE D,0
42300 POKE A,4
42400 VECTOR = PERE (49:41)
42500 IFAM = RANGE%(1,VECTOR)
42600 PRINT IFAM
42700 NEXT I
42800 VTAB 1
42900 NEXT J
43000 VTAB 10
43100 PRINT "X + 4,Y + 10"
43200 INPUT "(X,Y) "X,Y
43300 GOTO 41200
43400 RETURN
43500 REM
43600 REM
43700 REM
43800 REM
43900 REM
44000 REM
44100 REM
44200 REM
44300 REM
44400 REM
44500 REM
44600 REM
44700 REM
44800 REM
44900 REM
45000 REM
45100 REM
45200 REM
45300 REM
45400 REM
45500 REM
45600 REM
45700 REM
45800 REM
45900 REM
46000 REM
46100 REM
46200 REM
46300 REM
46400 REM
46500 REM
46600 REM
46700 REM
46800 REM
46900 REM
47000 REM
47100 REM
47200 REM
47300 REM
47400 REM
47500 REM
47600 REM
47700 REM
47800 REM
47900 REM
48000 REM
48100 REM
48200 REM
48300 REM
48400 REM
48500 REM
48600 REM
48700 REM
48800 REM
48900 REM
49000 REM
49100 REM
49200 REM
49300 REM
49400 REM
49500 REM
49600 REM
49700 REM
49800 REM
49900 REM
50000 REM
50100 REM
50200 REM
50300 REM
50400 REM
50500 REM
50600 REM
50700 REM
50800 REM
50900 REM
51000 REM
51100 REM
51200 REM
51300 REM
51400 REM
51500 REM
51600 REM
51700 REM
51800 REM
51900 REM
52000 REM
52100 REM
52200 REM
52300 REM
52400 REM
52500 REM
52600 REM
52700 REM
52800 REM
52900 REM
53000 REM
53100 REM
53200 REM
53300 REM
53400 REM
53500 REM
53600 REM
53700 REM
53800 REM
53900 REM
54000 REM
54100 REM
54200 REM
54300 REM
54400 REM
54500 REM
54600 REM
54700 REM
54800 REM
54900 REM
55000 REM
55100 REM
55200 REM
55300 REM
55400 REM
55500 REM
55600 REM
55700 REM
55800 REM
55900 REM
56000 REM
56100 REM
56200 REM
56300 REM
56400 REM
56500 REM
56600 REM
56700 REM
56800 REM
56900 REM
57000 REM
57100 REM
57200 REM
57300 REM
57400 REM
57500 REM
57600 REM
57700 REM
57800 REM
57900 REM
58000 REM
58100 REM
58200 REM
58300 REM
58400 REM
58500 REM
58600 REM
58700 REM
58800 REM
58900 REM
59000 REM
59100 REM
59200 REM
59300 REM
59400 REM
59500 REM
59600 REM
59700 REM
59800 REM
59900 REM
60000 REM
60100 REM
60200 REM
60300 REM
60400 REM
60500 REM
60600 REM
60700 REM
60800 REM
60900 REM
61000 REM
61100 REM
61200 REM
61300 REM
61400 REM
61500 REM
61600 REM
61700 REM
61800 REM
61900 REM
62000 REM
62100 REM
62200 REM
62300 REM
62400 REM
62500 REM
62600 REM
62700 REM
62800 REM
62900 REM
63000 REM
63100 REM
63200 REM
63300 REM
63400 REM
63500 REM
63600 REM
63700 REM
63800 REM
63900 REM
64000 REM
64100 REM
64200 REM
64300 REM
64400 REM
64500 REM
64600 REM
64700 REM
64800 REM
64900 REM
65000 REM
65100 REM
65200 REM
65300 REM
65400 REM
65500 REM
65600 REM
65700 REM
65800 REM
65900 REM
66000 REM
66100 REM
66200 REM
66300 REM
66400 REM
66500 REM
66600 REM
66700 REM
66800 REM
66900 REM
67000 REM
67100 REM
67200 REM
67300 REM
67400 REM
67500 REM
67600 REM
67700 REM
67800 REM
67900 REM
68000 REM
68100 REM
68200 REM
68300 REM
68400 REM
68500 REM
68600 REM
68700 REM
68800 REM
68900 REM
69000 REM
69100 REM
69200 REM
69300 REM
69400 REM
69500 REM
69600 REM
69700 REM
69800 REM
69900 REM
70000 REM
70100 REM
70200 REM
70300 REM
70400 REM
70500 REM
70600 REM
70700 REM
70800 REM
70900 REM
71000 REM
71100 REM
71200 REM
71300 REM
71400 REM
71500 REM
71600 REM
71700 REM
71800 REM
71900 REM
72000 REM
72100 REM
72200 REM
72300 REM
72400 REM
72500 REM
72600 REM
72700 REM
72800 REM
72900 REM
73000 REM
73100 REM
73200 REM
73300 REM
73400 REM
73500 REM
73600 REM
73700 REM
73800 REM
73900 REM
74000 REM
74100 REM
74200 REM
74300 REM
74400 REM
74500 REM
74600 REM
74700 REM
74800 REM
74900 REM
75000 REM
75100 REM
75200 REM
75300 REM
75400 REM
75500 REM
75600 REM
75700 REM
75800 REM
75900 REM
76000 REM
76100 REM
76200 REM
76300 REM
76400 REM
76500 REM
76600 REM
76700 REM
76800 REM
76900 REM
77000 REM
77100 REM
77200 REM
77300 REM
77400 REM
77500 REM
77600 REM
77700 REM
77800 REM
77900 REM
78000 REM
78100 REM
78200 REM
78300 REM
78400 REM
78500 REM
78600 REM
78700 REM
78800 REM
78900 REM
79000 REM
79100 REM
79200 REM
79300 REM
79400 REM
79500 REM
79600 REM
79700 REM
79800 REM
79900 REM
80000 REM
80100 REM
80200 REM
80300 REM
80400 REM
80500 REM
80600 REM
80700 REM
80800 REM
80900 REM
81000 REM
81100 REM
81200 REM
81300 REM
81400 REM
81500 REM
81600 REM
81700 REM
81800 REM
81900 REM
82000 REM
82100 REM
82200 REM
82300 REM
82400 REM
82500 REM
82600 REM
82700 REM
82800 REM
82900 REM
83000 REM
83100 REM
83200 REM
83300 REM
83400 REM
83500 REM
83600 REM
83700 REM
83800 REM
83900 REM
84000 REM
84100 REM
84200 REM
84300 REM
84400 REM
84500 REM
84600 REM
84700 REM
84800 REM
84900 REM
85000 REM
85100 REM
85200 REM
85300 REM
85400 REM
85500 REM
85600 REM
85700 REM
85800 REM
85900 REM
86000 REM
86100 REM
86200 REM
86300 REM
86400 REM
86500 REM
86600 REM
86700 REM
86800 REM
86900 REM
87000 REM
87100 REM
87200 REM
87300 REM
87400 REM
87500 REM
87600 REM
87700 REM
87800 REM
87900 REM
88000 REM
88100 REM
88200 REM
88300 REM
88400 REM
88500 REM
88600 REM
88700 REM
88800 REM
88900 REM
89000 REM
89100 REM
89200 REM
89300 REM
89400 REM
89500 REM
89600 REM
89700 REM
89800 REM
89900 REM
90000 REM
90100 REM
90200 REM
90300 REM
90400 REM
90500 REM
90600 REM
90700 REM
90800 REM
90900 REM
91000 REM
91100 REM
91200 REM
91300 REM
91400 REM
91500 REM
91600 REM
91700 REM
91800 REM
91900 REM
92000 REM
92100 REM
92200 REM
92300 REM
92400 REM
92500 REM
92600 REM
92700 REM
92800 REM
92900 REM
93000 REM
93100 REM
93200 REM
93300 REM
93400 REM
93500 REM
93600 REM
93700 REM
93800 REM
93900 REM
94000 REM
94100 REM
94200 REM
94300 REM
94400 REM
94500 REM
94600 REM
94700 REM
94800 REM
94900 REM
95000 REM
95100 REM
95200 REM
95300 REM
95400 REM
95500 REM
95600 REM
95700 REM
95800 REM
95900 REM
96000 REM
96100 REM
96200 REM
96300 REM
96400 REM
96500 REM
96600 REM
96700 REM
96800 REM
96900 REM
97000 REM
97100 REM
97200 REM
97300 REM
97400 REM
97500 REM
97600 REM
97700 REM
97800 REM
97900 REM
98000 REM
98100 REM
98200 REM
98300 REM
98400 REM
98500 REM
98600 REM
98700 REM
98800 REM
98900 REM
99000 REM
99100 REM
99200 REM
99300 REM
99400 REM
99500 REM
99600 REM
99700 REM
99800 REM
99900 REM

```



AD-A168 197

DEVELOPMENT OF ALGORITHMS AND HARDWARE FOR  
STRUCTURED-LIGHT VISION(U) GEORGE WASHINGTON UNIV  
WASHINGTON DC DEPT OF ELECTRICAL ENGN.

314

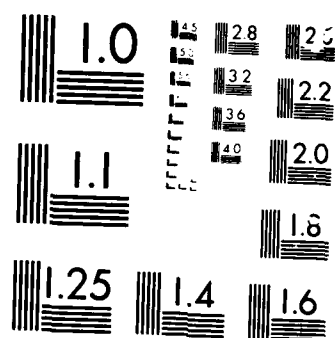
UNCLASSIFIED

M H LOEW ET AL. 1986 N00014-83-K-0578

F/G 20/6

142

[illegible]



MICROCOPY

10-11

~~183~~  
184

```

58000  REM
59000  G = .0000368
59100  REM
59200  FOR I = 0 TO 255
59300  RANGEZ(1,I) = (RA / (I + 1) * G)
59310  PRINT I,RANGEZ(1,I)
59400  NEXT I
59500  FOR I = 1 TO 3000: NEXT
60000  REM
60001  REM =====
60100  RETURN
61000  HOME
62001  PRINT "CPU DOUBLE ERROR HALT"
63002  VTAB 15
63003  END

```

1

APPENDIX E

S/W FOR OBJECT DESCRIPTION AND MATCHING

Computer vision system



**THE JOURNAL OF THE**

```

c*      compute the 2-D histogram of
c*      dz/dx and dz/dy
c*      load histogram to RAMTEK
c*      load separated plan to RAMTEK
c*
c*      -- histo
c*
c*      -- lo
c*      -- lo
c*
c*      -- plane
c*      -- lo
c*      -- edge
c*
c*      -- plane
c*      -- lo
c*      -- edge
c*
c*      find the individual plan
c*      equations, the edge coor of
c*      the plan and the node coor
c*
c*      -- matching
c*
c*      -- roan      rotate the ref. object
c*
c*      -- plane
c*      -- plane
c*      -- plane
c*
c*      -- sorting  rank plane equa. (object)
c*      -- sorting  rank plane equa. (view)
c*
c*
c*      object matching
c*
c*      rotate the ref. object
c*
c*      reference object plan
c*      forming
c*
c*      rank plane equa. (object)
c*      rank plane equa. (view)
c*
c*
c*      Major variables
c*      =====
c*      (i) related to specific view
c*      objj      256*256 integer array, range data
c*      idetx      256*256 integer array, dz/dx
c*      idety      256*256 integer array, dz/dy
c*      ihis      256*256 integer array, 2-D histogram
c*      ia        256*256 integer auxiliary array
c*      peq       5*5 real array, plane equa.
c*      ix        1000*2 integer array, edge coor.
c*      iy        1-D real array(size 1000), edge chain code
c*      c         30*3 real array, node coor.
c*      nopl      integer variable, # of planes
c*      nonode    integer variable, # of nodes
c*      nonob     integer variable, # of bottom nodes
c*
c*      (ii) related to reference object
c*      a         5*6*3 real array, node coor.
c*      ! ! ! --- 3-D coor.
c*      ! ! ! --- # of nodes
c*      ! ! ! --- # of positions
c*      pins      5*5 real array, plane equa.
c*      b         20*3 real array, node coor
c*      no_plane  integer variable, # of planes
c*      no_node   integer variable, # of nodes
c*      no_nodebo integer variable, # of bottom nodes
c*
c*

```



```

dimension pln(4,3),iq(10,2),g(3,4)
dimension ip(10),c(30,3),b(20,3),peq(5,5)
dimension ihs(256,256),ix(1000,2),iy(1000)
integer*2 iobj(256,256),idetx(256,256),idety(256,256)
integer*2 ia(256,256)

c
c call iniramfip(ierr)
c
c *****
c * read the data *
c *****
c
c call range_data(iobj)
c
c call lo(iobj,256,0,0,2,-20)
c call lo(iobj,256,0,512,2,-60)
c
c *****
c * compute first partial derivative *
c *****
c
c call differ(iobj,idetr,5,0,0)
c write(5,1111) ((idetx(i,j),i=120,145),j=45,175)
c format(26i3)
c call lo(idetr,256,960,512,10,128)
c
c call smoth(idetr)
c
c write(5,1111) ((idetr(i,j),i=120,145),j=45,175)
c
c call smoth(idetr)
c
c write(5,1111) ((idetr(i,j),i=120,145),j=45,175)
c
c call lo(idetr,256,320,0,15,128)
c call lo(idetr,256,320,512,10,90)
c
c call differ(iobj,idety,5,0,1)
c call smoth(idety)
c write(5,1111) ((idety(i,j),i=120,145),j=45,175)
c call smoth(idety)
c write(5,1111) ((idety(i,j),i=120,145),j=45,175)
c call lo(idety,256,640,0,1,5,128)
c call lo(idety,256,640,512,4,0,90)
c
c *****
c * subtract the background *
c *****
c
c call subback(iobj,idetr)
c call subback(iobj,idety)
c
c *****
c * find the 2D-histogram of gray level *

```

```

c *****
call histo(idetx, idety, ihis)
c -----
c to desply 2d-histogram
c -----
c do i=1,256
c do j=1,256
c   ie(i,j)=ihis(i,j)/30
c end do
c end do

c write(3,1112) ((ihis(i,j),i=125,140),j=110,170)
1112 format(16i3)

c call lo(ie,256,960,0.1,0)

c *****
c * find the equations of the planes *
c *****
c write(3,9994)
c =====
c find the no. of plane (i.e. the peak in the 2-D histo
c =====

      nopln=0
      do i=1,256
      do j=1,256
      if (ihis(i,j).lt 300) goto 60
      nopln=nopln+1
      iq(nopln,1)=i
      iq(nopln,2)=j
      end do
      end do

      nopln=nopln
      do i=2,nopln
      do j=1,i-1
      id=(iq(j,1)-iq(i,1))*2+(iq(j,2)-iq(i,2))*2
      if (id.gt 8) goto 70

      ie=ihis(iq(i,1),iq(i,2))-ihis(iq(j,1),iq(j,2))
      if (ie.gt 0) goto 75
      iq(i,1)=0
      iq(i,2)=0
      nopln=nopln-1
      goto 71

      iq(j,1)=0
      iq(j,2)=0
      nopln=nopln-1
      goto 71
      end do
      end do

      do i=1,nopln

```

```

72      if(iq(i,1) eq 0 and iq(i,2) eq 0) goto 72
       goto 73
       do i=1,nopln
         iq(i,1)=iq(i+1,1)
         iq(i,2)=iq(i+1,2)
       end do
73     end do

9998    nopln=nopln
       write(5,9998)nopln
       format(1x,'nopln = ',i4)

1113    do i=1,nopln
       write(5,1113) iq(i,1)-128,iq(i,2)-128,ihis(iq(i,1),iq(i,2))
       format(1x,'dz/dy,dz/dx,his -- ',3i10)
       end do
c=====
c find individual plane (i e. having same dz/dx and dz/dy)
c=====
c
       do i=1,256
       do j=1,256
         ia(i,j)=0
       end do
       end do

       do i=1,256
       do j=1,256
         m=midetx(i,j)+128
         n=midety(i,j)+128
         if(m.eq.255) goto 13
         do kk=i,nopln

13          if(m.ge.(iq(kk,1)-1) and m.le.(iq(kk,1)+1)
            2 ia(i,j)=kk*70

            end do
          end do
        end do

c          write(5,1111) ((ia(i,j),i=120,145),j=45,175)
c          call lo(ia,256,0,256,1,0)

c=====
c find plane equations
c=====
c
       do kk=i,nopln
       lp=0
       do i=1,256
       do j=1,256
         if (ia(i,j) eq 0) goto 23
         if (ia(i,j) ne kk*70) goto 23
         if (lp eq 1) goto 25
         lp=1
         pin(i,1)=float(i)
         pin(i,2)=float(j)

```

```

25         goto 23
        pln(2,1)=float(i)
23         pln(2,2)=float(j)
        end do
        end do

        lp=0
        iym=nint((pln(1,1)+pln(2,1))/2.0)
        do i=1,256
            if (ia(iym,1) ne kk*70) goto 26
            if (lp eq 1) goto 27
                lp=1
                xmin=i
                goto 26
                xmax=i
            end do

            idix1=abs(pln(1,2)-ixmin)
            idix2=abs(pln(1,2)-ixmax)

            if (idix1 gt idix2) goto 28
                pln(3,1)=float(iym)
                pln(3,2)=float(ixmax-2)
                goto 29
            pln(3,1)=float(iym)
            pln(3,2)=float(ixmin+2)

28
29         do i=1,3
            pln(i,3)=iobj(pln(i,1),pln(i,2))
        end do

        call plane(pln,d)

        peq(kk,2)=pln(4,1)
        peq(kk,3)=pln(4,2)
        peq(kk,4)=pln(4,3)
        peq(kk,5)=d

        write(5,9994)
        format(1x,'The plane equations which we finally find')
        write(5,30) kk,peq(kk,2),peq(kk,3),peq(kk,4),peq(kk,5)
        format(1x,'plane #',i1,'---',e12.5,' x',e12.5,' y',e12.5,
1' z= ',e12.5)

        end do

c *****
c * find individual plane image *
c *****

nn=0
do kk=1,nopl
do i=1,256
do j=1,256
    if (iobj(i,j) eq 255) goto 55
    di=float(i)*peq(kk,2)+float(j)*peq(kk,3)
    di=di+float(iobj(i,j))*peq(kk,4)-peq(kk,5)

```

```

      di=abs(d1)
      if(d1 gt 0.013) goto 55
      ia(i,j)=1
      goto 50

      ia(i,j)=0
      end do
      end do
      is=kk*320

c      write(5,1111) ((ia(i,j), i=120,145), j=45,175)
c      call lo(ia,256,15,256,200,0)

c      *****
c      * find node coordinates *
c      *****
c      call edge(ia,256,ix,iy,n)

c      do jk1=1,n
c      write(5,8181) ix(jk1,1),ix(jk1,2),iy(jk1,1)
      format(1x,'edge coord. and chain --',317)
      end do

c      =====
c      find the tuning points
c      =====
c      no=no+1
c      c(no,1)=float(ix(1,1))
c      c(no,2)=float(ix(1,2))

      iy(1)=iy(2)+iy(3)+iy(4)
      do i=3,n-4
      iy(i-1)=iy(i)+iy(i+1)+iy(i+2)
      iy=abs(iy(i-1)-iy(i-2))
      if (iy le 1) goto 90
      if (iy(i) eq 1 or iy(i) eq 8)
      * and (iy(i+1) eq 1 or iy(i+1) eq 8)
      * and (iy(i+2) eq 1 or iy(i+2) eq 8) goto 90

      no=no+1
      c(no,1)=float(ix(i+1,1))
      c(no,2)=float(ix(i+1,2))

90      end do
      end do

      do kj=1,no
      write(5,9995) kj,c(kj,1),c(kj,2)
      format(1x,'node #',12,' --',2f7.2)
      end do

9994      format(//)
c      =====

```

```

c      find the node coor
c      =====
      noo=no
      do i=2,no
        do j=1,i-1
          d=(c(j,1)-c(i,1))*2+(c(j,2)-c(i,2))*2
          if (d gt 0) goto 700
          c(j,1)=(c(j,1)+c(i,1))/2
          c(j,2)=(c(j,2)+c(i,2))/2
          c(i,1)=0
          c(i,2)=0
          noo=noo-1
          goto 771
        end do
      end do
      write(5,9994)
      write(5,9993) noo -- ',13)
      format(1x,'noo -- ',i3)
      do k=1,no
        write(5,9995) k,j,c(kj,1),c(kj,2)
      end do
      nonode=0
      do i=1,no
        if(c(i,1) eq 0 .and c(i,2) eq 0) goto 772
        nonode=nonode+1
        c(nonode,1)=c(i,1)
        c(nonode,2)=c(i,2)
      end do
      772
      write(5,9994)
      write(5,9996) nonode
      format(1x,'no of nodes ',i5)
      9996
      do i=1,nonode
        write(5,9999) c(i,1),c(i,2)
        format(1x,'nodes -- ',2e15.6)
      end do
      9999
      c      =====
      c      node ranking
      c      =====
      c      -----
      c      find the minimum y value in node coor.
      c      -----
      ymin=c(1,2)
      do i=2,nonode
        cd=c(i,2)-ymin
        if (cd gt 0) goto 120
        ymin=c(i,2)
      end do
      120
      c      write(5,9991) ymin
      9991
      format(1x,'ymin -- ',f7.0)

```

```

c
c      separate the nodes as bottom node and non-bottom nodes
c      n1 --- bottom
c      n2 --- non-bottom
c

```

```

      nonob=0
      n1=0
      n2=9
      do i=1,nonode
        cd=c(i,2)-ymin
        if(cd.gt.2) goto 122
        nonob=nonob+1
        n1=n1+1
        b(n1,1)=c(i,1)
        b(n1,2)=c(i,2)
        goto 121
      n2=n2+1
      b(n2,1)=c(i,1)
      b(n2,2)=c(i,2)
    end do

```

122

121

```

      write(5,9994)
      write(5,9990) nonob
      format(1x,'nonob -- ',i5)

```

9990

```

c
c      rank the codes in such order :
c      (i) bottom nodes first
c      (ii) from left to right
c

```

```

      do i=1,n1-1
        m=i
        ii=i+1
        sm=b(i,1)
        do j=ii,n1
          if(sm.le.b(j,1)) goto 130
          sm=b(j,1)
          m=j
        end do
      do j=1,2
        t=b(i,j)
        b(i,j)=b(m,j)
        b(m,j)=t
      end do
    end do

```

130

```

      do i=1,n1
        c(i,1)=b(i,1)
        c(i,2)=b(i,2)
      end do

```

```

      do i=10,n2-1
        m=1

```

```

11=i+1
sm=b(i,1)

do j=1,n2
  if(sm.le.b(j,1)) goto 131
  sm=b(j,1)
  m=j
end do

do j=1,2
  t=b(i,j)
  b(i,j)=b(m,j)
  b(m,j)=t
end do

end do

do i=n1+1,n2-9+n1
  ii=i-n1+9
  c(i,1)=b(ii,1)
  c(i,2)=b(ii,2)
end do

do i=1,nnode
  ii=nint(c(i,1))
  jj=nint(c(i,2))
  c(i,3)=iobj(ii,jj)
end do

write(5,9994)
do i=1,nnode
  write(5,9989) c(i,1),c(i,2),c(i,3)
  format(1x,'ranked nodes --',3f8.0)
end do

9989
c *****
c * object matching *
c *****
c
c call matching(nopln,nnode,nnode,c,peq,icode)
end

```





```

SUBROUTINE iniramrjp(ierr)
C
C  init ramtek fip levels 3 & 4
C  NOTE  this initialization assumes files std003 ini & std004 ini
C         are in [ee208] directory
C
C
C  input & output are 5 & 6 respectively
C
C  CHARACTER*1 yn
C  CALL lev3ini(ierr)
C  CALL lev4ini(ierr)
C  WRITE(5,100)
100  FORMAT('GRAMTEK full reset (answer y/n/Y/N) ?= ')
101  READ(5,101)yn
C  FORMAT(a1)
C  IF(yn EQ 'Y' OR yn EQ 'y')THEN
C    CALL rset(ierr)
C  END IF
C  CALL lovlr(ierr)
C  RETURN
C  END

```

[illegible]

|         |       |     |     |         |            |          |       |           |       |
|---------|-------|-----|-----|---------|------------|----------|-------|-----------|-------|
| RRRRRRR | AAAAA | NN  | NN  | GGGGGGG | EEEEEEEEEE | DDDDDDDD | AAAAA | TTTTTTTTT | AAAAA |
| RRRRRRR | AAAAA | NN  | NN  | GGGGGGG | EEEEEEEEEE | DDDDDDDD | AAAAA | TTTTTTTTT | AAAAA |
| RR      | AA    | NN  | NN  | GG      | EE         | DD       | AA    | TT        | AA    |
| RR      | AA    | NN  | NN  | GG      | EE         | DD       | AA    | TT        | AA    |
| RR      | AA    | NNN | NN  | GG      | EE         | DD       | AA    | TT        | AA    |
| RR      | AA    | NNN | NN  | GG      | EE         | DD       | AA    | TT        | AA    |
| RRRRRRR | AA    | NN  | NN  | GG      | EEEEEEEE   | DD       | AA    | TT        | AA    |
| RRRRRRR | AA    | NN  | NN  | GG      | EEEEEEEE   | DD       | AA    | TT        | AA    |
| RR      | AAAAA | NN  | NNN | GGGGG   | EE         | DD       | AAAAA | TT        | AA    |
| RR      | AAAAA | NN  | NNN | GGGGG   | EE         | DD       | AAAAA | TT        | AAAAA |
| RR      | AA    | NN  | NN  | GG      | EE         | DD       | AA    | TT        | AA    |
| RR      | AA    | NN  | NN  | GG      | EE         | DD       | AA    | TT        | AA    |
| RR      | AA    | NN  | NN  | GGGGG   | EEEEEEEE   | DDDDDD   | AA    | TT        | AA    |
| RR      | AA    | NN  | NN  | GGGGG   | EEEEEEEE   | DDDDDD   | AA    | TT        | AA    |
| RR      | AA    | NN  | NN  | GGGGG   | EEEEEEEE   | DDDDDD   | AA    | TT        | AA    |
| RR      | AA    | NN  | NN  | GGGGG   | EEEEEEEE   | DDDDDD   | AA    | TT        | AA    |

[illegible]

File \_DUAL [FACGAD VISIONRANGE\_DATA FOR.21 (15769.58;0)], last revised on 18-SEP-1985 09:28, is a B block sequential file owned by UIC [EECS,FACGAD] The records are variable length with implied (CR) carriage control The longest record is 61 bytes

Job RANGE\_DATA (9) queued to SYS\$PRINT on 9-DEC-1985 15:23 by user FACGAD, UIC [EECS,FACGAD], under account EECS at priority 4, started on printer \_LPAD on 9-DEC-1985 15:32 from queue LPAD

[illegible]

```

subroutine range_data(iobj)
dimension pln(4,3), p(3,4)
integer*2 iobj(256,256)

do i=1,256
  do j=1,256
    iobj(i,j)=255
  end do
end do

c
c *****
c   find the plane equ.
c *****
c
c   plane #1 -- A1,A2,A3,A4
c
c
  pln(1,1)=49.76
  pln(1,2)=150.
  pln(1,3)=98.214
  pln(2,1)=134.852
  pln(2,2)=110.
  pln(2,3)=42.928
  pln(3,1)=141.244
  pln(3,2)=50.
  pln(3,3)=30.
  call plane(pln,d)

do i=1,3
  p(1,i)=pln(4,i)
end do
p(1,4)=d

c
c   plane #2 -- A4,A5,A6
c
c
  pln(1,1)=49.76
  pln(1,2)=150.
  pln(1,3)=98.214
  pln(2,1)=134.852
  pln(2,2)=110.
  pln(2,3)=42.928
  pln(3,1)=137.674
  pln(3,2)=170.
  pln(3,3)=119.818
  call plane(pln,d)

do i=1,3
  p(2,i)=pln(4,i)
end do

```

$$p(2, 4) = d$$

plane #3 -- A2, A3, A5, A6

$p1n(1, 1) = 141$ . 244  
 $p1n(1, 2) = 50$ .  
 $p1n(1, 3) = 30$ .

$\text{pin}(2, 1) = 134.852$   
 $\text{pin}(2, 2) = 110.$   
 $\text{pin}(2, 3) = 42.928$

$\text{pin}(3, 1) = 137.674$   
 $\text{pin}(3, 2) = 170.$   
 $\text{pin}(3, 3) = 119.818$

```
call plane(pln,d)
```

```
do i=1,3
  p(3,i)=p1n(4,i)
end do
p(3,4)=d
```

```

1      write(5,1)
      format(1x,'plane equa. to generate ori. data')
      do i=1,3
2      write(5,2) i,p(1,1),p(1,2),p(1,3),p(1,4)
      format(1x,'plane #',11,'---',e12.5,'x',e12.5,'y',
      *e12.5,'z',e12.5)
      end do

```

```

c *****
c find the range data
c *****
c

```

Plane #1

```

do i=20,50
  jj=nint(3.360*i-17.2)
  do j=50,jj
    iobj(i,j)=nint((p(1,4)-p(1,1)*i-p(1,2)*j)/p(1,3))
  end do
end do

```

```

do i=50, 135
  jj=nint(-0.470*i+173.392)
  do j=50, jj
    iobj(i, j)=nint((p(1, 4)-p(1, 1)*i-p(1, 2)*j)/p(1, 3))
  end do
end do

```

```

do i=135,141
  jj=nint(-9 387*1+1375 82)
  do j=50,jj
    10b(j,i,j)=nint((p(1,4)-p(1,1)*1-p(1,2)*j)/p(1,3))
  end do
end do

```

```

end do

c
c
c
-----
plane #2
-----

do i=50,135
  j1=nint(-0.47*i+173.392)
  j2=nint(0.227*i+138.680)
  do j=j1, j2
    iobj(i, j)=nint((p(2,4)-p(2,1)*i-p(2,2)*j)/p(2,3))
  end do
end do

do i=135,138
  j1=nint(21.26*i-2757.16)
  j2=nint(0.227*i+138.680)
  do j=j1, j2
    iobj(i, j)=nint((p(2,4)-p(2,1)*i-p(2,2)*j)/p(2,3))
  end do
end do

c
c
c
-----
plane #3
-----

do i=135,137
  j1=nint(-9.387*i+1375.82)
  j2=nint(21.26*i-2757.16)
  do j=j1, j2
    iobj(i, j)=nint((p(3,4)-p(3,1)*i-p(3,2)*j)/p(3,3))
  end do
end do

do i=138,141
  j1=nint(-9.387*i+1375.82)
  j2=nint(-6.34*i+1042.82)
  do j=j1, j2
    iobj(i, j)=nint((p(3,4)-p(3,1)*i-p(3,2)*j)/p(3,3))
  end do
end do

do i=141,157
  j1=nint(-6.34*i+1042.82)
  do j=50, j1
    iobj(i, j)=nint((p(3,4)-p(3,1)*i-p(3,2)*j)/p(3,3))
  end do
end do

return
end

```



```

subroutine lo(iobj,n,ix,iy,scale,ioffset)
integer*2 iobj(256,256),ia(256,256)

do i=1,256
  do j=1,256
    ia(i,j)=iobj(j,257-i)*scale+ioffset
  end do
end do

call lobs(ia,256,ix,iy,ierr)

return
end

```



[illegible][illegible]

|            |        |         |        |        |
|------------|--------|---------|--------|--------|
| FFFFFFFFFF | 000000 | RRRRRRR | 1111   | 333333 |
| FFFFFFFFFF | 000000 | RRRRRRR | 1111   | 333333 |
| FF         | 00     | RR      | 1111   | 33     |
| FF         | 00     | RR      | 1111   | 33     |
| FF         | 00     | RR      | 1111   | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FFFFFFF    | 00     | RRRRRRR | 1111   | 33     |
| FFFFFFF    | 00     | RRRRRRR | 1111   | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 00     | RR      | 11     | 33     |
| FF         | 000000 | RR      | 111111 | 333333 |
| FF         | 000000 | RR      | 111111 | 333333 |
| FF         | 000000 | RR      | 111111 | 333333 |

File\_DUAL [FACGAD VISION]DIFFER FOR:13 (13601,7,0), last revised on 11-SEP-1985 10:51, is a 2 block sequential file owned by UIC [EECS,FACGAD] The records are variablelength with implied (CR) carriage control. The longest record is 51 bytes.

Job DIFFER (1) queued to SYS\$PRINT on 9-DEC-1985 15:23 by user FACGAD, UIC [EECS,FACGAD], under account EECS at priority 4, started on printer\_LPAO on 9-DEC-1985 15:29 from queue\_LPAO

```
CCCCCCCCC0 CCCCCCCC01 CCCCCCCCCC
```

||||| Digital Equipment Corporation - VAX/VMS Version V4.1 |||||

```
CCCCCCCCC0 CCCCCCCCCC CCCCCCCCCC
```

```

subroutine differ(iobj, idet, scale, id)
integer*2 iobj(256,256), idet(256,256)

if (id eq 1) goto 100

do i=1,256
  do j=1,255
    idet(i,j)=iobj(i,j+1)-iobj(i,j)
    if (abs(idet(i,j)) lt 20) goto 1
    idet(i,j)=idet(i,j-1)
    goto 2
  end do
  idet(i,j)=idet(i,j)*scale
end do

do i=1,256
  idet(i,256)=idet(i,255)
end do

return

100  do j=1,256
    do i=1,255
      idet(i,j)=iobj(i+1,j)-iobj(i,j)
      if (abs(idet(i,j)) lt 20) goto 3
      idet(i,j)=idet(i-1,j)
      goto 4
    end do
    idet(i,j)=idet(i,j)*scale
  end do

do i=1,256
  idet(256,i)=idet(255,i)
end do

return
end

```

[illegible]

|       |      |      |      |      |     |
|-------|------|------|------|------|-----|
| FFFFF | AAA  | CCCC | GGGG | AAA  | 000 |
| F     | A    | C    | G    | A    | 0   |
| FF    | A    | C    | G    | A    | 0   |
| FFF   | A    | C    | G    | A    | 0   |
| F     | AAAA | C    | G    | AAAA | 0   |
| F     | A    | C    | G    | A    | 0   |
| F     | A    | CCCC | GGG  | A    | 000 |

[illegible]

|            |        |          |      |        |
|------------|--------|----------|------|--------|
| FFFFFFFFFF | 000000 | RRRRRRRR | 1111 | 000000 |
| FFFFFFFFFF | 000000 | RRRRRRRR | 1111 | 000000 |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 00     | RR       | RR   | 00     |
| FFFFFFFFFF | 00     | RRRRRRRR | 1111 | 00     |
| FFFFFFFFFF | 00     | RRRRRRRR | 1111 | 00     |
| FF         | 00     | RR       | RR   | 0000   |
| FF         | 00     | RR       | RR   | 0000   |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 00     | RR       | RR   | 00     |
| FF         | 000000 | RR       | RR   | 000000 |
| FF         | 000000 | RR       | RR   | 000000 |
| FF         | 000000 | RR       | RR   | 000000 |

File\_DUAL [FACGAD VISION]SMOOTH FOR.10 (6740.22.0), last revised on 21-SEP-1985 16:59, is a 2 block sequential file owned by UIC [EECS.FACGAD] The records are variable length with implied (CR) carriage control. The longest record is 63 bytes.

Job SMOTH (12) queued to SYS@PRINT on 9-DEC-1985 15:23 by user FACQAD, UIC [EECS,FACQAD], under account EECS at priority 4, started on printer LPA0 on 9-DEC-1985 15 29 from queue LPA0.

[illegible]

```

subroutine smoth(idetr)
integer*2 idetr(256,256), idety(256,256)

do i=2,255
  do j=2,255
    ka=idetr(i-1,j-1)+idetr(i-1,j)+idetr(i-1,j+1)
    kb=idetr(i,j-1)+idetr(i,j)+idetr(i,j+1)
    kc=idetr(i+1,j-1)+idetr(i+1,j)+idetr(i+1,j+1)
    idety(i,j)=nint(float(ka+kb+kc)/9)
  end do
end do

do i=1,256
  do j=1,256
    idetr(i,j)=idety(i,j)
  end do
end do

return
end

```



```

subroutine subback(iobj, idet)
integer*2 iobj(256,256), idet(256,256)
*****
c subtract the background
*****
do i=1,256
do j=1,256
if (iobj(i,j) ne 255) goto 20
idet(i,j)=255
end do
end do

return
end

```

20

[illegible][illegible][illegible]

File \_DUA1 [FACGAD VISIONHISTO FOR:12 (17244.26.Q), last revised on 12-SEP-1985 11.11, is a 2 block sequential file owned by UIC [EECS.FACGAD]. The records are variable length with implied (CR) carriage control. The longest record is 49 bytes.

Job HISTO (14) queued to SYS@PRINT on 9-DEC-1985 15:24 by user FACQAD, UIC [EECS,FACQAD], under account EECS at priority 4, started on printer\_LPA0: on 9-DEC-1985 15:30 from queue LPA0.

[illegible]

```

subroutine histo(idetx, idety, ihis)
integer*2 idetx(256,256), idety(256,256)
dimension ihis(256,256)

do i=1,256
  do j=1,256
    ihis(i,j)=0
  end do
end do

do i=1,256
  do j=1,256
    if(idetx(i,j) eq 255) goto 1
    ii=idetx(i,j)+128
    jj=idety(i,j)+128
    ihis(ii,jj)=ihis(ii,jj)+1
  end do
end do

return
end

```





```

subroutine plane(pln,d)
dimension pln(4,3)

a1=pln(2,1)-pln(1,1)
b1=pln(2,2)-pln(1,2)
c1=pln(2,3)-pln(1,3)

a2=pln(3,1)-pln(1,1)
b2=pln(3,2)-pln(1,2)
c2=pln(3,3)-pln(1,3)

d1=b1*c2-b2*c1
d2=a2*c1-a1*c2
d3=a1*b2-a2*b1

d=d1*pln(1,1)+d2*pln(1,2)+d3*pln(1,3)
write(5,1) a1,b1,c1
format(1x,'a1,b1,c1 --','3e15.6)

c
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2
```



```

Subroutine edge(ipic,m,ix,iy,n)
integer*2 ipic(256,256)
dimension ia(8),ix(1000,2),iy(1000)

do i=1,1000
    iy(i)=0
    ix(i,1)=0
    ix(i,2)=0
end do

c *****
c search the boundary
c *****
c =====
c      search the start point
c =====
c      do i=1,m
c          do j=1,n
c              if (ipic(i,j).eq.1) goto 41
c          end do
c      end do

41      io=i
         jo=j
         write(5,3) i,j
         format('start point at (' ,i3' , 'i3, ')')
         =====
         go clockwise
         =====
         icha=2
         n=1
         ix(n,1)=i
         ix(n,2)=j
         iy(n)=icha

         write(5,4) n,icha
         format('n=' ,i3,' direction ',i5)

         if (n.gt.1.and.i.eq.io.and.j.eq.jo) goto 1000
         n=n+1
         ia(1)=ipic(i-1,j+1)
         ia(2)=ipic(i,j+1)
         ia(3)=ipic(i+1,j+1)
         ia(4)=ipic(i+1,j)
         ia(5)=ipic(i+1,j-1)
         ia(6)=ipic(i,j-1)
         ia(7)=ipic(i-1,j-1)
         ia(8)=ipic(i-1,j)
         if (icha.eq.1) then
             if (ia(7).eq.1) goto 111
             if (ia(8).eq.1) goto 112
             if (ia(1).eq.1) goto 113
             if (ia(2).eq.1) goto 114
             if (ia(3).eq.1) goto 115
             if (ia(4).eq.1) goto 116
             icha=7
             i=i-1
             j=j-1
             goto 100
         else
             goto 111
         endif
     end do
end subroutine edge

```

```

112      1cha=8
        1=1-1
        goto 100
113      1cha=1
        1=1-1
        J=J+1
        goto 100
114      1cha=2
        J=J+1
        goto 100
115      1cha=3
        1=1+1
        J=J+1
        goto 100
116      1cha=4
        1=1+1
        goto 100
        else if (1cha eq 2) then
          if (1a(1) eq 1) goto 211
          if (1a(2) eq 1) goto 212
          if (1a(3) eq 1) goto 213
          if (1a(4) eq 1) goto 214
          if (1a(5) eq 1) goto 215
211      1cha=1
        1=1-1
        J=J+1
        goto 100
212      1cha=2
        J=J+1
        goto 100
213      1cha=3
        1=1+1
        J=J+1
        goto 100
214      1cha=4
        1=1+1
        goto 100
215      1cha=5
        1=1+1
        J=J-1
        goto 100
        else if (1cha eq 3) then
          if (1a(1) eq 1) goto 311
          if (1a(2) eq 1) goto 312
          if (1a(3) eq 1) goto 313
          if (1a(4) eq 1) goto 314
          if (1a(5) eq 1) goto 315
          if (1a(6) eq 1) goto 316
311      1cha=1
        1=1-1
        J=J+1
        goto 100
312      1cha=2
        J=J+1
        goto 100
313      1cha=3
        1=1+1
        J=J+1
        goto 100
314      1cha=4

```

```

315      i=i+1
      goto 100
      1cha=5
      i=i+1
      j=j-1
316      goto 100
      1cha=6
      j=j-1
      goto 100
      else if (1cha eq 4) then
      if (1a(3).eq 1) goto 411
      if (1a(4).eq 1) goto 412
      if (1a(5).eq 1) goto 413
      if (1a(6).eq 1) goto 414
      if (1a(7).eq 1) goto 415
      1cha=3
411      i=i+1
      j=j+1
      goto 100
412      1cha=4
      i=i+1
      goto 100
413      1cha=5
      i=i+1
      j=j-1
      goto 100
414      1cha=6
      j=j-1
      goto 100
415      1cha=7
      i=i-1
      j=j-1
      goto 100
      else if (1cha eq 5) then
      if (1a(3).eq 1) goto 511
      if (1a(4).eq 1) goto 512
      if (1a(5).eq 1) goto 513
      if (1a(6).eq 1) goto 514
      if (1a(7).eq 1) goto 515
      if (1a(8).eq 1) goto 516
      1cha=3
511      i=i+1
      j=j+1
      goto 100
512      1cha=4
      i=i+1
      goto 100
513      1cha=5
      i=i+1
      j=j-1
      goto 100
514      1cha=6
      j=j-1
      goto 100
515      1cha=7
      i=i-1
      j=j-1
      goto 100
516      1cha=8
      i=i-1

```

```

goto 100
else if (icha eq 6) then
  if (ia(5) eq 1) goto 611
  if (ia(6) eq 1) goto 612
  if (ia(7) eq 1) goto 613
  if (ia(8) eq 1) goto 614
  if (ia(1) eq 1) goto 615
  icha=5
  i=i+1
  j=j-1
goto 100
  icha=6
  j=j-1
goto 100
  icha=7
  i=i-1
  j=j-1
goto 100
  icha=8
  i=i-1
goto 100
  icha=1
  i=i-1
  j=j+1
goto 100
else if (icha eq 7) then
  if (ia(5) eq 1) goto 711
  if (ia(6) eq 1) goto 712
  if (ia(7) eq 1) goto 713
  if (ia(8) eq 1) goto 714
  if (ia(1) eq 1) goto 715
  if (ia(2) eq 1) goto 716
  icha=9
  i=i+1
  j=j-1
goto 100
  icha=6
  j=j-1
goto 100
  icha=7
  i=i-1
  j=j-1
goto 100
  icha=8
  i=i-1
goto 100
  icha=1
  i=i-1
  j=j+1
goto 100
  icha=2
  j=j+1
goto 100
else if (icha eq 8) then
  if (ia(7) eq 1) goto 811
  if (ia(8) eq 1) goto 812
  if (ia(1) eq 1) goto 813
  if (ia(2) eq 1) goto 814
  if (ia(3) eq 1) goto 815
  icha=7

```

611

612

613

614

615

711

712

713

714

715

716

811

```

      i=i-1
      j=j-1
      goto 100
      ica=8
      i=i-1
      goto 100
      ica=1
      i=i-1
      j=j+1
      goto 100
      ica=2
      j=j+1
      goto 100
      ica=3
      i=i+1
      j=j+1
      goto 100
      end if

      n=n-1
      return
      end

```

812

813

814

815

1000





```

subroutine roent(a1,a3,b1,b3,angle)
c
20  write(5,20) a1,a3,b1,b3
    format(1x,'a1,a3,b1,b3 -- ',4f7.1)

    a=sqrt(a1**2+a3**2)
    b=sqrt(b1**2+b3**2)

    x=b3/b
    fi=asin(x)
    y=a3/a
    angle=asin(y)

c
1  write(5,1) x,fi
    format(1x,'x,asin(x) -- ',2e15.6)
c
2  write(5,2) y,angle
    format(1x,'y,asin(y) -- ',2e15.6)

    angle=-angle-fi
c
3  write(5,3) angle
    format(1x,'angle -- ',e15.6)

    b=-b*asin(angle)+b3*cos(angle)
    d=abs(a3-b)

    if (d.lt.3.0) goto 10
        angle=3.1415926/2-angle
    b=-b*asin(angle)+b3*cos(angle)
    d=abs(a3-b)

    if (d.lt.3.0) goto 10
        write(5,5)
        format(1x,'Fail to find the rotation angle')

    return
end
10
5

```

APPENDIX F

RELEVANT TECHNICAL PAPERS WRITTEN DURING THE COURSE OF THIS WORK

PIPE<sup>1</sup>: A SPECIALIZED COMPUTER ARCHITECTURE FOR ROBOT VISION.

Ernest W. Kent

Sensory-Interactive Robotics Group

The National Bureau of Standards<sup>2</sup>

1. PIPE is a registered trademark of Digital/Analog Design Associates, Inc. of New York.
2. This work is the product of United States Government employees, and is not subject to U.S. copyright.

PIPE (Pipelined Image Processing Engine), is an experimental, multi-stage, multi-pipelined image processing device. It can acquire images from a variety of sources, such as analog or digital television cameras, ranging devices, and conformal mapping arrays. It can process sequences of images in real time, through a series of local neighborhood and point operations, under the control of a host device. Its output can be configured for monitors, robot vision systems, iconic to symbolic mapping devices, and image processing computers. In addition to a forward flow of images through successive stages of operations as in a traditional pipeline, other paths between the stages of the device permit concurrent, interacting pipelining of image flow in other directions. In particular, recursive paths returning images into each stage, and feedback of the results of operations from each stage to the preceding stage are supported. The architecture facilitates a variety of functions such as relaxation and interactions of images over time. Numerous operations are supported; within each stage these include arithmetic and Boolean neighborhood and point operations on images. Between-stage operations on each pixel include thresholding, Boolean and arithmetic operations, functional mappings, and a variety of functions for combining pixel data converging via the multiple pipelined image paths. The device also implements alternative processing modes, including "MIMD" operations specific to regions of interest defined by the host device or by previous operations on the image, and variable resolution pyramid operations.

## Requirements of Robot Vision.

A robot vision system has requirements which differ in some respects from those of other types of machine vision. Since the function of a robot is to perform physical actions in space and time similar to those performed by humans, it is not surprising that these requirements are similar to those imposed on biological vision systems. In some machine vision applications, such as interpretation of images from diagnostic or satellite equipment, each new picture is a new problem; in robot vision, like biological vision, the input is a stream of images in which each frame differs from the last by some small displacement of the camera or of the objects in the scene. In many machine vision tasks the important function is to recognize certain kinds of objects, but in robot vision, again like biological vision, most of the vision system's time is spent providing information about the position of things in space for guidance and servoing. This is true after objects have already been recognized, before they are recognized, and even if they cannot be recognized by the system. The robot must understand the spatial occupancy of its environment and its own relation to it in order to avoid striking surfaces, whether or not those surfaces are part of recognized objects. Ideally the robot vision system should provide a description of the geometric properties of unrecognized objects sufficient to permit the robot to manipulate them; for example, to remove a foreign object from the workspace. Most importantly, a robot vision system shares with the biological system the necessity of synchronization with real-time events.

From these considerations, we can derive some important properties of a robot vision system. Because the robot's function is oriented towards actions and events which extend over time, the appropriate unit of analysis is the image sequence rather than a single image. This implies that the system must capture and operate on image sequences of a length appropriate to the speed of events. Because the robot's world possesses continuity, the system should take advantage of information discovered in previous views. This suggests that the system should not only build internal models of the environment from successive views, but also that it should be able to make use of hypotheses from this model in interpreting subsequent images. The importance of servoing over classification implies that the system must provide information at many levels of analysis. That is, it may be required to supply information on range to points, on inclinations of edges and surfaces, on translation and rotation velocities of features, and similar descriptive properties of objects in space and time. These must be made available to the

robot control system as rapidly as they are discovered, prior to and independently of classification. Above all, the system must operate in real-time; a late answer is no answer for a robot guidance system. It is this which compels us to examine special-purpose architectures as a means for accomplishing the other criteria.

### The NBS Robot Vision System

The general plan of a robot system which incorporates all of these functions is easy enough to sketch, and the basic ideas are included in Figure 1, which is adapted from a robot system being developed by the Sensory-Interactive Robotics Group at NBS. On the left is a sensory processing hierarchy, in the center a knowledge representation system, and on the right a task-decomposition and control hierarchy. This entire system could be considered a "special purpose architecture", in the sense that it is a special purpose machine which has computers as components. However, most of the elements are currently implemented on individual computers of traditional design, and in this chapter I intend to focus primarily on some particular vision elements within this system which, for reasons of required processing speed, have been implemented on special purpose architectures in the sense of specially-designed hardware.

Before considering these elements in detail, a brief review of the system operations will help clarify their tasks and design goals. The sensory processing hierarchy accepts data from the sensors (in particular cameras for vision sensing) and attempts to form a hierarchical syntactic description of the current view of the world. Note that symbolic and parametric information is made collaterally available to the system's knowledge representation at every level of this process. The task-decomposition and control hierarchy on the opposite side accepts goals in the form of external commands. It attempts to use generic knowledge of how to choose actions together with particular knowledge of the current state of the world to generate actions which accomplish the goals. Knowledge of the current state of the world is made collaterally available to every level of the task-decomposition and control hierarchy from the system knowledge representation.

The knowledge representation contains much more than the information currently available from the sensory processing hierarchy. In addition to information which comes from a priori knowledge sources, and knowledge of the control hierarchy's current state, the knowledge representation contains a description of the world built up over all past views from the

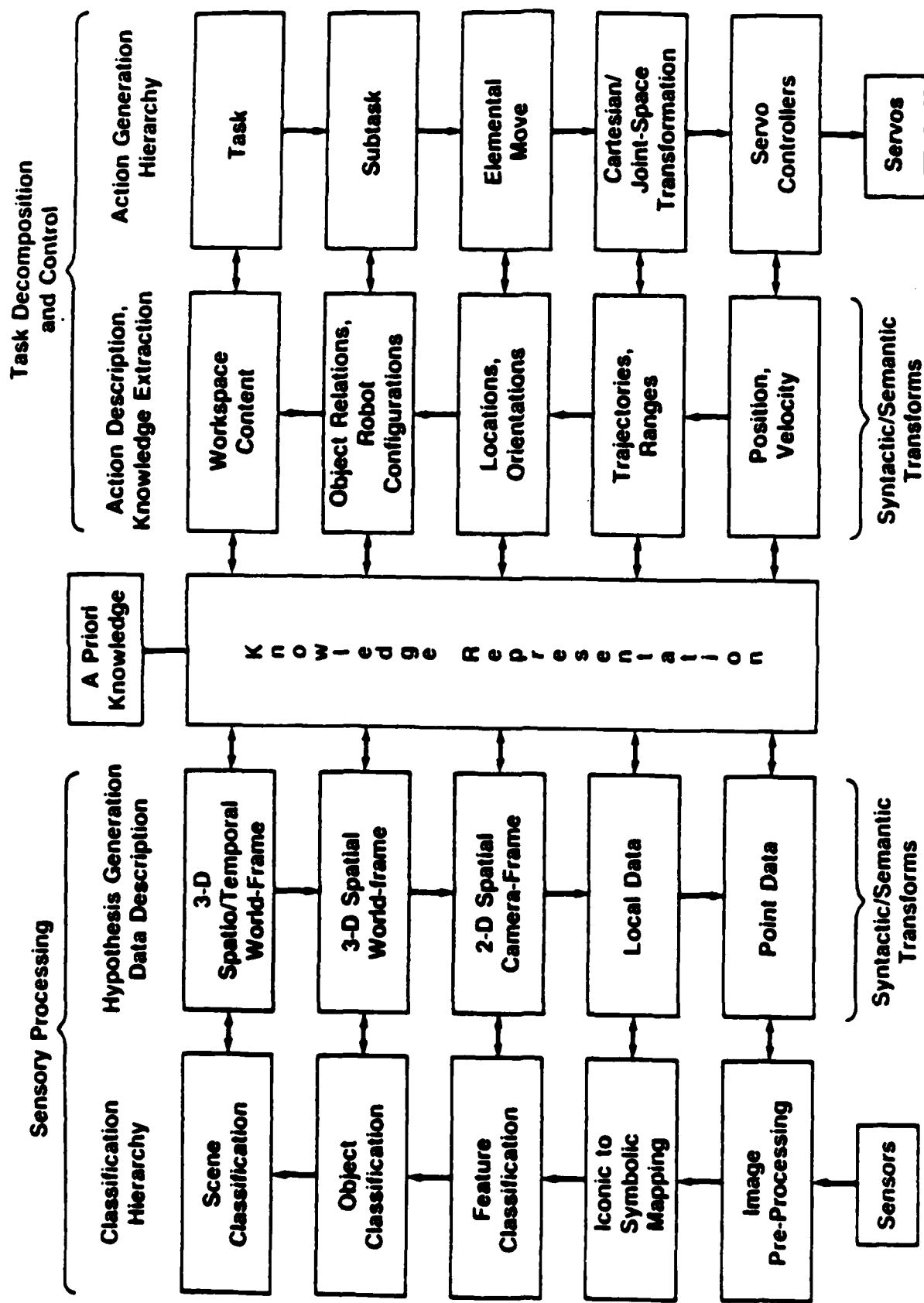


Figure 1. The elements of the NBS image-processing configuration in relation to the NBS robot system.



camera. It thus contains much information which is not extractable from the current viewing window, but which bears on the interpretation of that which is. The current scene descriptions from the sensory processing hierarchy must be reconciled with this information, and the sensory processing hierarchy in turn may be guided by it. Similarly, the knowledge representation's model of the world must be formulated into specific answers to any particular instantaneous requirement of the task-decomposition and control hierarchy. In turn, the state of the action-generating processes provides guidance for the knowledge representation in interpreting its input from the sensory system. There is thus a two-way flow between the knowledge representation and the sensory processing system which "serves" the internal model of the world to the observed data, and which provides hypotheses that guide the processing and interpretation of the data. A similar reciprocal flow links the knowledge representation's internal model of the world with the questions and information produced by the control hierarchy.

The dimension along which the control hierarchy is divided into functional levels (task, subtask, elemental cartesian move, joint-space velocity) is not the same as the dimension along which the vision processing hierarchy is divided into levels (image-plane point properties, image-plane features, world-frame objects.) The knowledge representation, which is not itself hierarchically structured, contains information relevant to all of the different descriptions of the world required by these levels. Its information is maintained in a semantic form from which syntactic and parametric descriptions of the world adapted to the requirements of any of the functional modules can be produced. The representation scheme of this semantic form is chosen principally for convenience in maintaining and organizing the information. The syntactic forms of the various levels of the sensory and control systems are chosen primarily for functional utility in the modules concerned. The "syntactic/semantic transform" modules extract level-specific information from the knowledge representation and instantiate it into the frame and symbols needed for any given functional module. In the other direction, they contain systems for incorporating level-specific syntactic and parametric sensory or control information into the internal model.

This diagram outlines a general conception of the relations of sensory and other knowledge in a robot system. What is vastly more difficult is to specify the algorithms and hardware required to give substance to the boxes in such a diagram. A first generation system was constructed entirely from micro-computers. It used binary image processing of both normal and structured-light images. The knowledge representation and the range of

requests acceptable from the control hierarchy were very limited in scope. Nonetheless, this system could accept CAD descriptions of simple parts, recognize them, and provide servo information that allowed the control system to manipulate them in real-time. A second generation system has been under development for some time. This project includes much more elaborate structures for knowledge representation, and a correspondingly richer sensory processing capability which employs gray-scale vision. Most of this second generation system development effort consists of software improvements still running in a multi-microcomputer dataflow (as opposed to distributed) system. With the move to gray-scale vision, however, it became apparent that the early stages of image processing involved computations which could not be handled in real-time by microcomputers. Further, it was felt that for these early image-processing functions a sufficiently good understanding of the requirements of robot vision existed to justify the development of special-purpose vision architectures. Accordingly, an image preprocessor (called PIPE, for Pipelined Image Processing Engine) and a feature extractor (called ISMAP for Iconic to Symbolic MAPper) were designed. The first prototypes of these devices are now in operation. Their relations are depicted in Figure 2, which shows the data-flow paths for the PIPE and ISMAP elements, to and from the upper levels of the system (indicated here as "host memory").

These devices will replace microcomputers as the first two elements of the sensory processing hierarchy in the second generation of the robot system depicted in figure 1. The first of these elements is an initial processing stage (image pre-processing) which accepts an image consisting of an array of gray-scale values and produces a similar array of symbolic values which encode features of the gray-scale array, such as edges, or textures. This process of feature detection produces an iconic description of image features from an iconic description of image intensities. That is, for both the input and the output of this process, the global geometric relations of the input values or the output features are implicit in their location in the image array. The feature detection process transforms one iconic array into the other by making local properties explicit as symbols. In most cases the process attempts to describe local properties (features) which achieve some independence of circumstances such as illumination.

Following the feature detection process, the second element of the system performs a feature extraction step (iconic to symbolic mapping.) This begins the process of producing a relational feature description of objects which is independent of circumstances of viewing position. In this step, the iconic image of features, in which symbol values are indexed by location, is

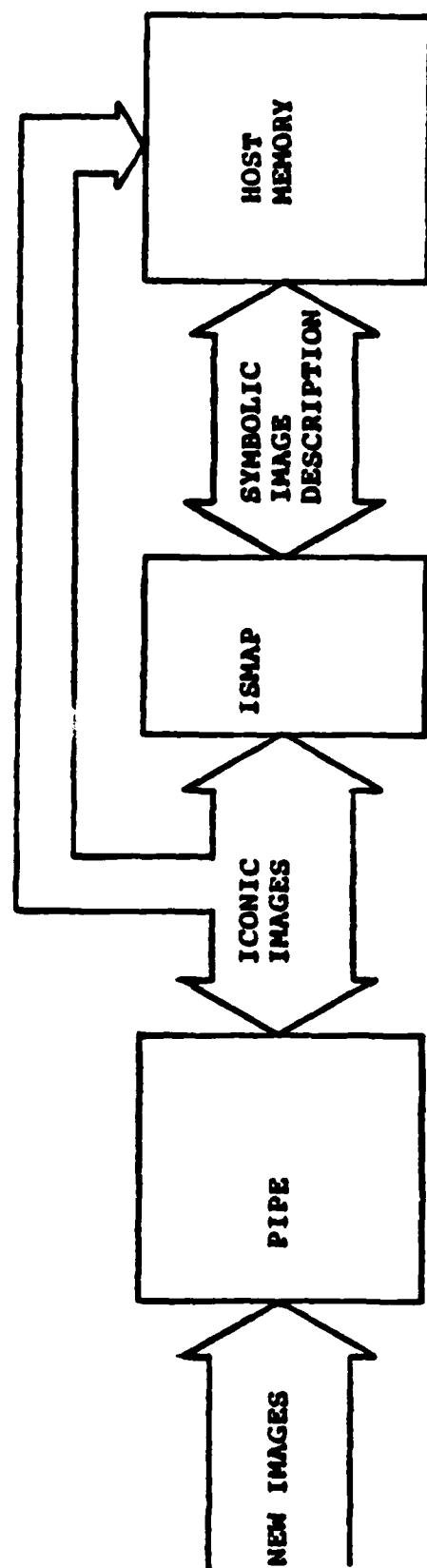


Figure 2. Major Image-Flow relationships between PIPE, ISMAP, and the host devices.

mapped into a space in which location is indexed by feature value. This produces a representation in which the data which was previously explicit (the feature type) is now implicit in the location of the data in the representation. The image location of the features, which was previously implicit in the iconic representation, is now the explicit data represented by values in the new space. This accomplishes two goals; it permits finding the locations of features of interest by direct indexing rather than by exhaustively searching the image, and it extracts locations as data to which subsequent operations can be applied. These subsequent operations attempt to find meaningful relations among the location data which correspond to geometric relations of features. The geometric relations discovered then serve as input to subsequent classification processes.

### Design Philosophy of PIPE

In designing the PIPE and ISMAP devices to play these roles in the system, we attempted to optimize the design for the special requirements of robot vision discussed above. The principal goals selected were: 1) real-time processing of images at field-rate, 2) provision for interactions between related images, such as those arising from dynamic image sequences or from stereoscopic views, 3) provision of the ability to apply different algorithms to different regions of the image in real time, 4) ability to perform multi-resolution image processing, and 5) provision for guiding processing by knowledge-based commands and "hypothesis images" supplied from the upper levels of the system.

PIPE is a hardware device specialized for parallel image processing rather than a fully general purpose parallel computer. Through its design, it facilitates a variety of common and important image-processing techniques, as well as several experimental approaches. Within the broad limits of the processes it supports, it is an extremely fast and flexible device. On the other hand, it is not a general purpose computer and it is not possible to program arbitrary algorithms on it, or at least not in an efficient manner. In most cases we have found that processes which are well suited to PIPE may be substituted for others which are not, while accomplishing the same image-processing goal. PIPE is intended as a processor for local operations on images; it is not designed to perform efficiently operations that require global knowledge of the image. It is intended that PIPE operate in conjunction with a host, such as the upper levels of the NBS robot vision system, which will perform global image operations, relieved of the processing burden of large scale repetitive local operations. Thus, PIPE itself accepts iconic data images, and typically produces iconic

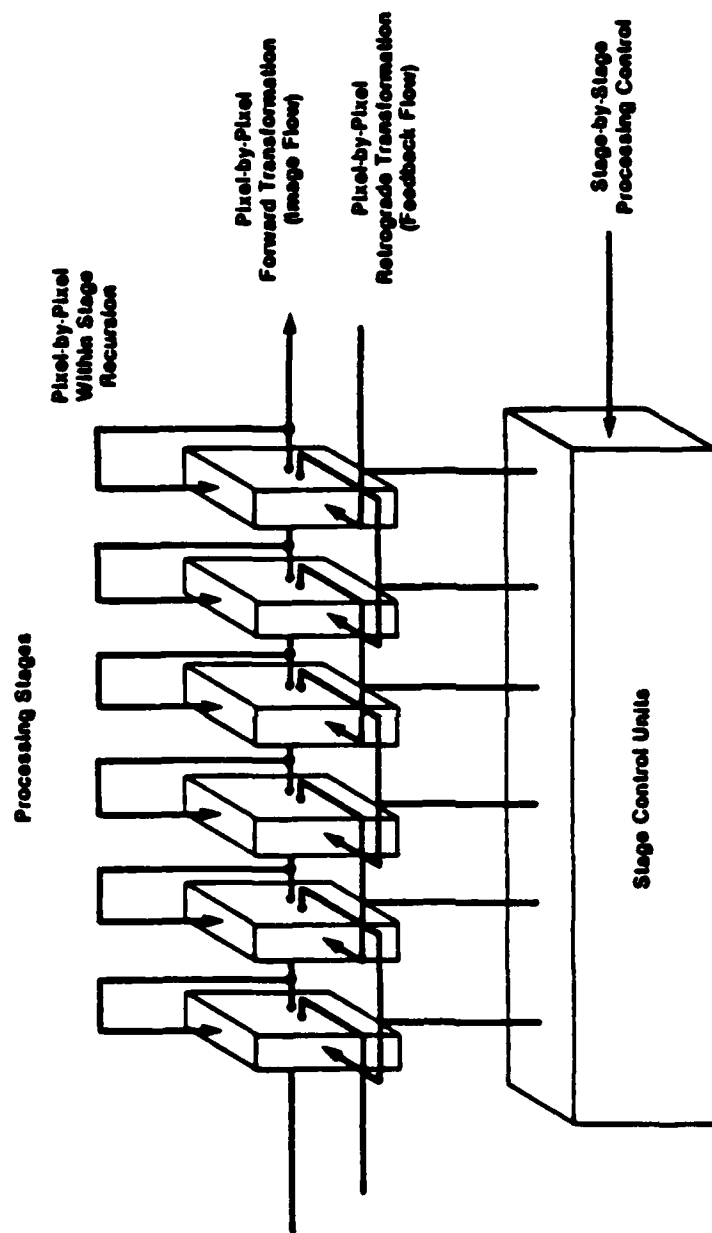
images whose pixel values are Boolean vectors describing local properties of the pixel neighborhood. PIPE relieves the upper levels of the system of costly low-level local processing which must be performed over the entire image space.

The basic design of PIPE was partly inspired by an earlier proposed machine<sup>1,2</sup> which was in turn adapted from biological models of image processing. The essential organization is a three-dimensional architecture consisting of a sequence of image-processing planes (Figure 3.) Each processing plane has storage arrays which receive images, and operators which act on them. The storage arrays may be considered to be in-register with one another in the third dimension, so that each pixel neighborhood in an array occupies a step in a pipeline of processes extending up through the stack of planes. The result is an image-flow architecture in which the images move upwards through the stack as subsequent images replace them from below. At each stage the operations on the pixels provide interaction in the lateral directions with neighbors in the same array, and in the vertical directions with neighbors in the preceding and succeeding arrays. Logically the machine can be considered as a bundle of bidirectional pipeline processors with lateral interactions.

In practice, PIPE actually consists of stages with storage arrays and computational modules (figure 4.) which operate over every pixel neighborhood in the arrays in a single field time (1/60 sec.) Images are transferred from stage to stage at field-rate (60 images/sec) by three concurrent pathways which provide for interacting, image-flow transfers between stages. These interconnect a variable number of identical modular image-processing stages. The three pathways, shown in figure 3, are: the forward pathway, which acts as a traditional pipelined image-processing path; the retrograde pathway, which carries images in the opposite direction (i.e., from the output of a stage to the input of its predecessor); and the recursive pathway, which carries an image from the output of a stage back into the input of the same stage.

At the input to each stage (labeled "combining logic" in figure 4.) the images carried by the three pathways may be subjected individually to any arithmetic or Boolean operation. Any linear arithmetic or Boolean operation may then be used to combine them into a final input image, prior to its storage in one of two buffers within the stage.

Within each stage, the computational operators may act on images stored in either or both of the two buffers. These operators are contained in the sections marked "operations" in figure 4. Each stage can perform two simultaneous and independent



**Figure 3.** Major Image-Flow connections between processing stages in PIPE.

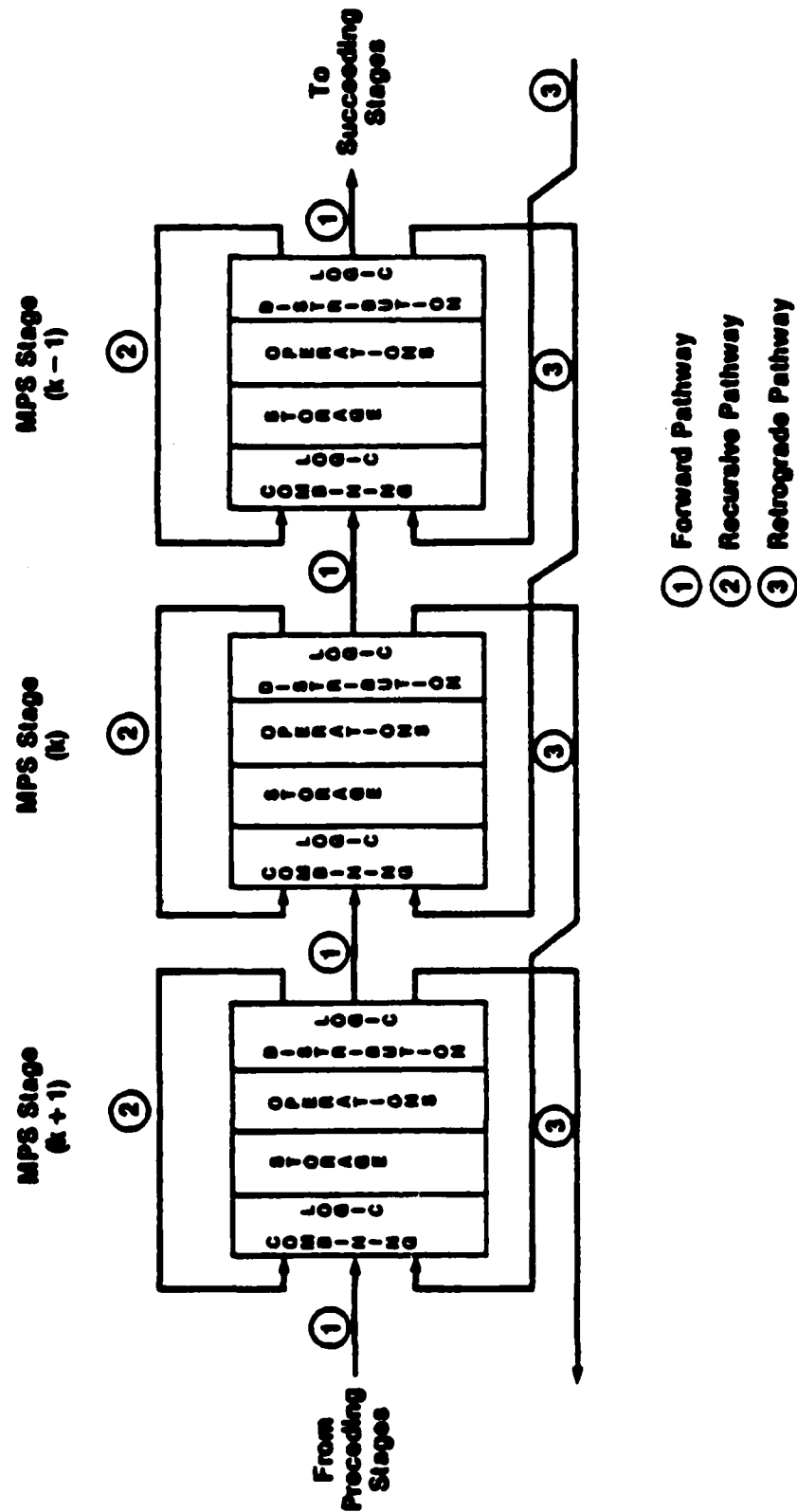


Figure 4. Image-Flow connectivity in relation to the major elements of a hardware PIPE stage.

arithmetic or Boolean neighborhood operations. Following the neighborhood operations, and prior to output from the stage, the images resulting from the neighborhood operations, or either of the images in the buffers, may undergo a further transformation by an arbitrary function of one or two arguments. If the function is of two arguments, the second argument may be drawn from any source within the stage, including the transformations of same or the other image. The forward, recursive, and retrograde pathways, in any combination, may accept images from the result of any of these operations or from any of the buffers. This is controlled by the section marked "distribution logic" in figure 4.

The representation of figure 4 shows the actual physical grouping of processing elements and storage as they exist in the machine for structural reasons. Functionally, the combining logic of one stage and the operations and distribution sections of the preceding stage may form a more convenient conceptual processing unit for many purposes. Figure 5 shows the architecture of PIPE redrawn in this fashion, where the processors are the conceptual rather than the physical units. In this figure the frame-buffer storage elements can be unbundled from the processing units, so that the tri-directional image-flow relations are easier to visualize.

In an alternative mode, one of the two image buffers in each stage may serve as a map for selecting the processing algorithms being applied to the contents of the other buffer. In this mode, PIPE functions as multi-instruction stream multi-data stream (MIMD) machine, with one buffer defining regions of interest over which each set of algorithms shall be applied, on a pixel-by-pixel basis, as the image is processed. In the prototype version, sixteen such alternative processing algorithms may be selected for different regions of the image within a single field time. However, this is easily increased (up to 256 processing algorithms) simply by adding additional storage to each stage to hold the appropriate tables and parameters.

The third mode permits PIPE to function as a multi-resolution pyramid machine. In this mode, the images carried by the forward pathway are reduced in size by one half at each stage, while sizes of the images carried by the retrograde pathway are doubled at each stage. The images carried by the recursive pathway remain unchanged in resolution. Any combination of stages may operate in this mode, under program control.

In addition to the pathways mentioned, PIPE contains four "wild card" busses which may be used to transport images from higher levels of processing, or from any buffer in the machine,



into any other buffer in the machine. It also has two special stages, for input and output, which communicate with the sensors, with processors in upper levels of the system, or with special auxiliary devices. DMA channels allow video display or streaming input and output from any buffer in the machine.

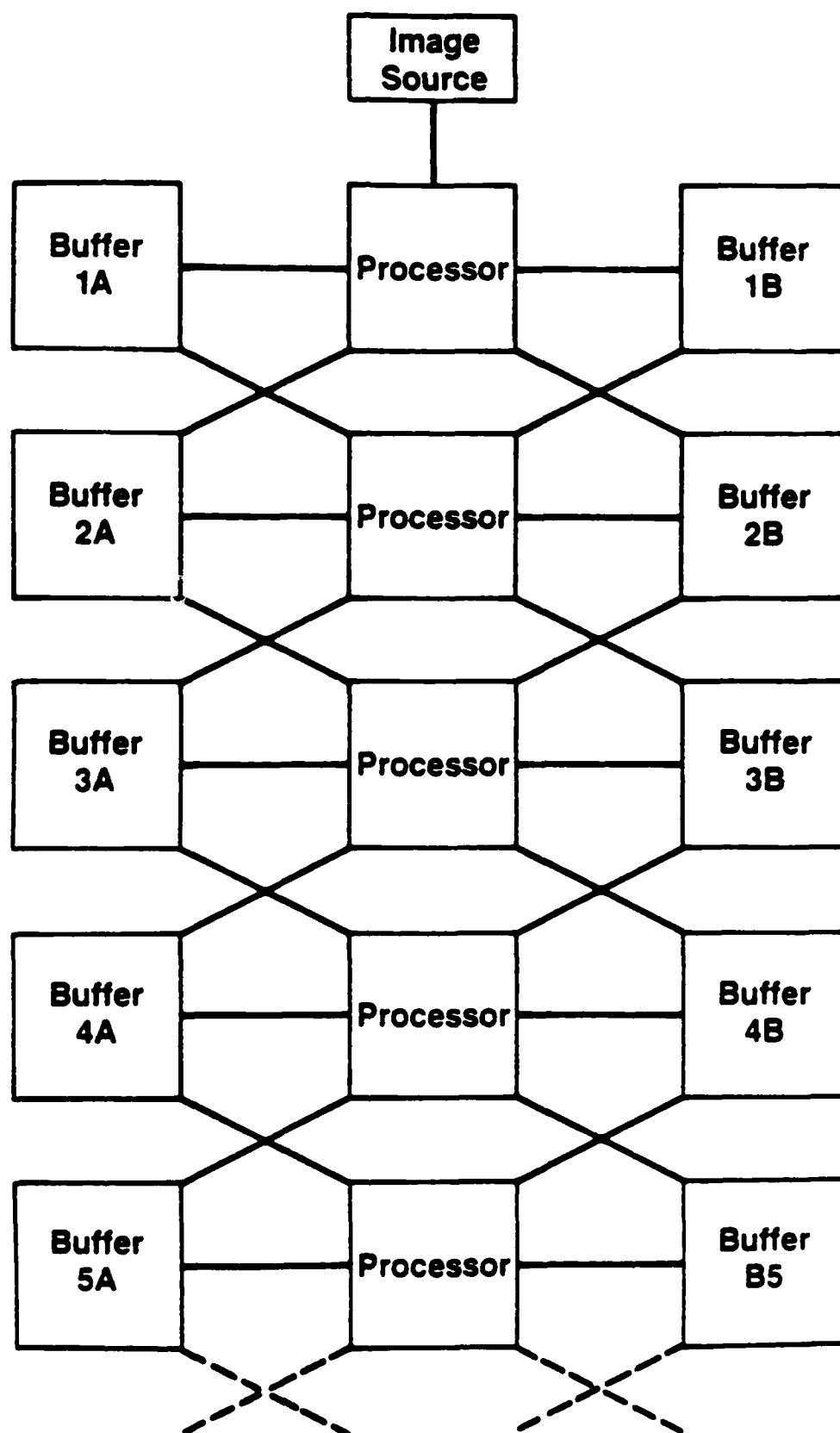
In any of PIPE's operating modes, the operations of every stage are completely independent, and can be completely reconfigured in the inter-field time by an associated stage-sequencer control unit, which in turn may select stage configurations from a stored sequence, or on command from the higher levels.

#### Operation of PIPE

The initial version of PIPE consists of a sequence of identical image processing stages, sandwiched between a special input processor and a special output processor. The input processor accepts an image from any device that encodes two-dimensional images. It serves as a buffer between the rest of the image processing stages and the outside world. Each successive processing stage receives image data in an identical format, operates on it, and passes it on to the next stage for further processing. This sequence is repeated every television field-time. When an image emerges at the far end of the sequence, it is processed by the special output stage and presented to upper levels of the robot vision system or to a host computer.

The image processing stages between the input and output stages are all identical and interchangeable, but can each perform different operations on the image sequences that they encounter. Usually, each image processing stage receives three input images and transmits three output images. The input images arrive from the processing stage immediately behind each stage, from the processing stage immediately ahead, and from a result of the preceding operation performed by the image processing stage itself. Similarly, the results of processing a current image are transmitted by each stage to the next processing stage in the sequence, to the immediately-preceding processing stage, and recursively back into the image processing stage itself. These three outputs are usually not identical, and each may furnish part or all of the inputs to other stages for the subsequent step in processing. The three inputs may be weighted and combined in each image processing stage, in any fashion, before they are processed.

In addition to these input and output paths, the four 'wildcard' paths may be sources or destinations for input and output. Unlike the three principal paths connecting the stages,



**Figure 3.** Image-Flow connectivity in relation to conceptual functional processors of PIPE.

these wildcard paths are common to all stages, so that only one stage can write to a particular wildcard path at a time, but any or all stages can accept input from them. The wildcard paths allow images to be moved arbitrarily between stages, instead of having to step through from stage to stage. There are no restrictions on the number of destinations for an image output to a wildcard path.

Although there are physically only three pathways between stages (excluding the "wildcard" busses), every pixel neighborhood in an image is processed and sent over these paths in every field time. The result is that PIPE simulates a fully parallel image-flow machine; each pixel appears to have a real-time, private line to an homologous pixel processor in three target stages. There are numerous reasons for requiring the three input and output paths from each image processing stage. It is clear that the forward path allows a chain of operations to be performed on sequential images, giving rise in real time to a transformed image stream (with a constant delay). Similarly, the recursive path allows a pipeline of arbitrary length to be simulated by any stage. It also facilitates the use of algorithms that perform many iterations before converging to a desired result (e.g., relaxation algorithms, or the simulation of large neighborhood operators by successive applications of smaller neighborhood operators). The path to the preceding image processing stage allows operations to be performed using temporal as well as spatial neighborhoods. It also allows information inserted at the output stage by the upper levels of the system to participate in the processing directly. This, for example, allows expectations or image models to be used to guide the processing at all levels, on a pixel-by-pixel basis.

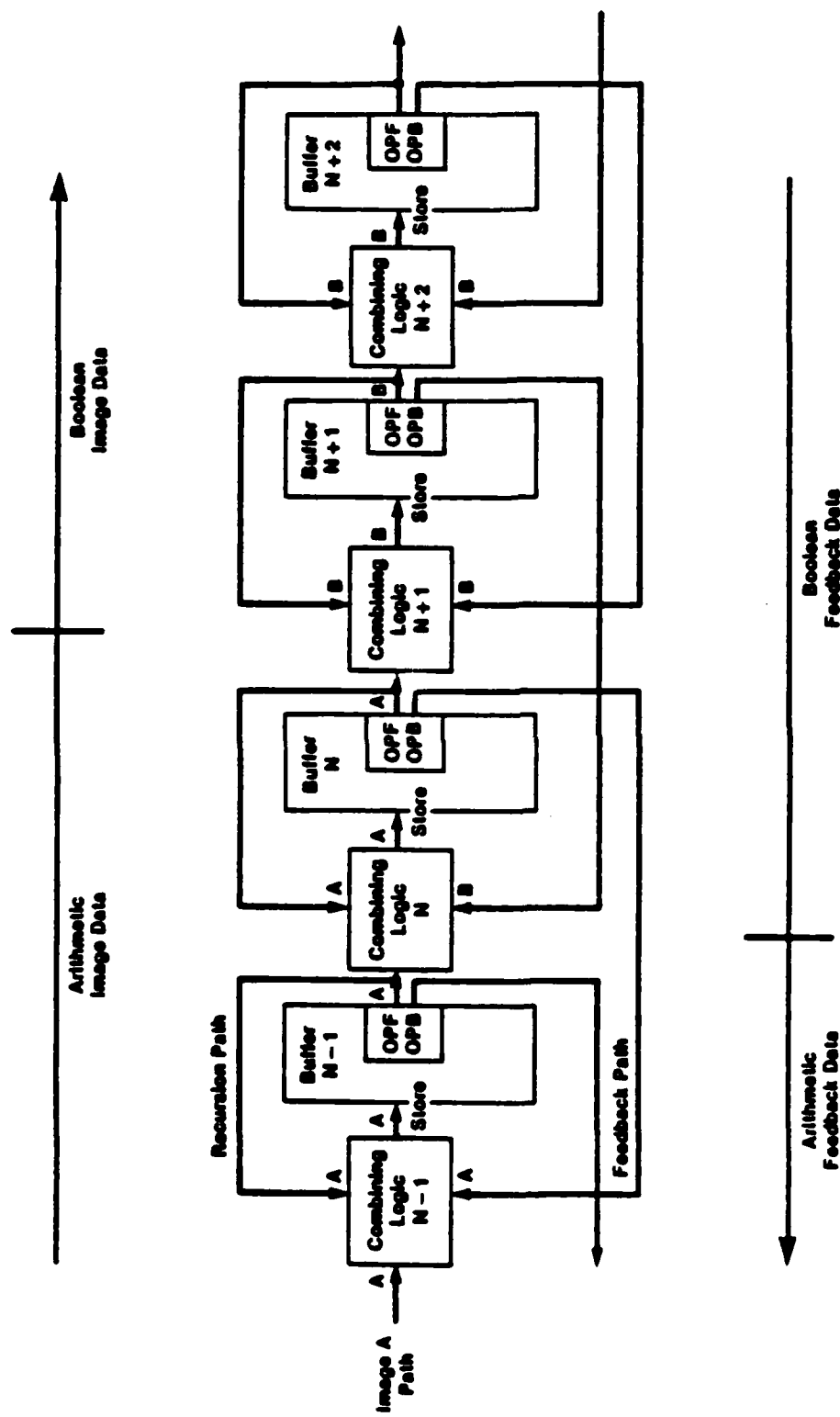
It is helpful in understanding the functions of these processing pathways to consider each in isolation first. If only the forward input path is operative (i.e., the combining weights for the retrograde and recursion paths are set to zero), we have a simple image pipeline processor which can sequentially apply a variety of neighborhood operators to the series of images flowing through it. It can perform either arithmetic or Boolean neighborhood operations and, by thresholding, convert an arithmetic image into a Boolean image. For example, it might be used to smooth an arithmetic gray scale image, apply edge detection operators to it, threshold the "edginess" value to form a binary edge image and then apply Boolean neighborhood operations to find features in the edges. The operation types and parametric values for these operations would be set individually for each stage by the stage control units, which in turn would be instructed (for example from the upper levels of the system) via the input marked "stage-by-stage processing control" in Figure 3.

A second single-path case results if both the forward and retrograde paths' combining functions are zero. Assume that images had previously been loaded into all the processing stages. The recursive path would then cause the image field in each stage to pass through the forward or backward transformation operation recursively, while the images "marched in place". A variety of relaxation operations can be implemented in this way.

For the final single path case, consider that the weights assigned to the forward and recursive paths are zero, leaving only the retrograde pathway active. When the set of such paths is considered in isolation, it becomes clear that it forms a processing chain that is a retrograde counterpart of the forward pipeline. It would, in fact be possible to select appropriate retrograde transformations, insert fields of data at the back of the device, process them through to the front, and get the same result as running the system in the normal direction. The purpose of this is not to provide a bidirectional image processor, but to permit input (at the "output" end of the device) of synthesized images. Such images influence the processing of the normally flowing images by direct interaction, and correspond to "expectancies", "models", "hypotheses", or "attention functions."

The retrograde images are not only able to affect processing of the forward images, but are affected themselves by interaction with them. (The effects that the two image sequences exert on each other may be different because the neighborhood operators on the forward and backward paths are independent). Retrograde images will usually be generated by knowledge-based processes in higher level components of the robot vision system. They may initially appear in Boolean form, but, as shown in Figure 6, provision is made for all four possible combinations of arithmetic and Boolean inputs and outputs in the combining logic between stages. This permits a descending Boolean image to be instantiated into arithmetic image values by interaction with the ascending arithmetic image. This occurs in the same stage in which the ascending arithmetic image representation is thresholded to become a Boolean image (Stage "N" of figure 6.) Both the ascending data image and the descending "hypothesis" image can pass across this interface. A major function of PIPE will be to explore the effectiveness of various approaches to hypothesis-guided iconic image processing.

There are a great many ways in which various combinations of these pathways can be used in processing by operating on combinations of processed images arriving over the three pathways. I will consider only a few illustrative examples.



**Figure 6.** Modes of interaction between arithmetic and Boolean images.

If the preceding and succeeding image fields are considered to contain future and past instances of a field, respectively (as is true in a dynamic image), then forward path corresponds to a path from the future, recursion to a path from the present, and the retrograde path to one from the past. The weighted sum of the three paths' contents the forms a convolution operation on the temporal neighborhood of a pixel. This may occur at the same time as a spatial neighborhood convolution operation is being performed on the contemporary spatial neighborhood in each stage, so that a combined spatio-temporal convolution is achieved.

Boolean information can be processed in an interesting way by combining the outputs of the forward operator from the previous stage and the recursive input from the current stage. Consider the case of a single stage treated in this fashion for eight field-times, using "SHIFT recursive then OR recursive with forward" as the combining operation. Let the incoming images from the previous stage (forward path) have Boolean values resulting from thresholding of eight successive and independent feature detection operations such as oriented edge detection, texture measures, etc. The second stage will accumulate images from the eight preceding Boolean operations into an image composed of eight-bit Boolean vectors, each bit representing the presence or absence of an independent image property at that location. Subsequent Boolean neighborhood operations may then apply independent operators to each bit plane of a neighborhood of such vectors.

#### HARDWARE DETAILS OF THE PIPE STAGES.

##### Input Stage

A special input stage is used to capture and buffer images from input devices. This allows PIPE to accept digital or analog signals from any device using standard RS-170 television signals and timing. Either of two selectable analog signals is digitized by an eight-bit real-time digitizer. The input stage is capable of acquiring a digitized image of 256 x 240 pixels while remaining synchronized with RS-170 signals. Alternatively, it can capture 256 x 256 pixel images from non-RS-170 signals while internally employing non-standard pixel rates. It can continually capture such images at standard television field rates, and place them in either of the two field buffers contained in the input stage. While storing an image into one of these buffers the input stage can also simultaneously store an image into either buffer,

such as a difference image, formed by an ALU operation between the incoming image and a previously captured image. The contents of either of the buffers in the input stage can be sent to the first of the processing stages, while the next image is being acquired.

PIPE accepts eight-bit input data, and this precision is maintained throughout the machine. Intermediate arithmetic operations within subsequent stages are carried to sufficient precision to insure no loss of accuracy when the result is rounded to eight bits for transmission to subsequent stages. The data may be treated as either unsigned eight-bit numbers, or as two's complement signed numbers with the high bit indicating the sign, or as boolean vectors of eight independent bits. All stages may independently select the representation employed, so that unsigned input data may be processed until an operation which generates negative values occurs, treated as signed data thereafter until a thresholding operation occurs, and then treated as Boolean data.

### Processing Stages

Following the Input stage, there are a series of modular processing stages (MPS). The MPSs are the "stages" referred to in the preceding sections, and are the elements which perform most of PIPE's processing. All MPSs are of identical modular construction, and are physically interchangeable simply by switching circuit boards. Thus, any MPS can operate at any position in the processing chain, and the processing chain can have a variable length. Eight MPSs are employed for the present development phase of PIPE. The block diagram of a MPS is portrayed in figure 7.

The pre-storage input section of the Nth MPS accepts three eight-bit 256 x 256 pixel images as input. These come from the forward output of the N-1st MPS, from the recursive output of the operation performed on the previous contents of the Nth MPS, and from the retrograde output of the N+1st MPS. Each data stream may consist, independently of the other two, of arithmetic or Boolean (eight-bit Boolean vector) data, but a given data stream entering a MPS must be entirely Boolean or arithmetic within any single image field.

Before generating a final eight-bit image from the three data streams, the input section of each MPS performs an arbitrary table-lookup transformation on each them independently and simultaneously (forward, backward, and recursive L.U.T. in figure 7.) Each of the three lookup tables is one of three sets of 32 tables selectable by the stage operation micro-instruction. The

resulting three Boolean and/or arithmetic data streams are then combined through independently-programmable full-function ALUs into a single arithmetic or Boolean data stream (figure 7, ALU-A and ALU-B.) This data stream is then used to load either of the two selectable field buffers within the MPS (buffer A and buffer B in the figure.) Alternatively, either or both buffers can be filled using the wildcard busses or by direct DMA from the host. The contents of both of these field buffers are then available to subsequent operations of the MPS. External device access to the data in these buffers is also available; an external device may read from or write into either buffer in a random access manner at 400,000 pixels/sec., with auto-indexed addressing supported on command. The wildcard busses provide streaming access to external devices (including monitors) at pixel rates.

The hardware that implements the output functions of the MPS, subsequent to the field buffer storage step, is physically contained on a separate circuit card to allow it to be replaced with other special functional modules, should this be desirable. This circuitry is represented by the area to the right of the frame buffers in Figure 7. For neighborhood operations, an eight-bit image is selected by reading the contents of one of the two field buffers in the MPS. The image is transformed by a single-valued mapping through one of 32 program-selectable lookup-tables (pre-nop L.U.T.), and the pixels of the resulting image are passed to two neighborhood operators (NOP A, and NOP B), of which there are two kinds. The first type of neighborhood operator is an arithmetic convolution operation, while the second is a Boolean operation. For either operator, the neighborhood of operation is (at present) 3 x 3 pixels square, and the operation is accomplished in 200 nsec. Pixel neighborhoods are generated by passing the data stream through a 3-line buffer.

In the arithmetic case, the convolution operation uses arbitrary positive or negative eight-bit neighborhood weights, and maintains twelve-bit accuracy in its intermediate results. The final eight-bit arithmetic result bus is produced by non-biased rounding, from a 20-bit sum. This insures that no loss of precision occurs within a stage due to arithmetic underflow or overflow. The full eight bit precision of the input is thus maintained between stages throughout the machine. In the Boolean case, the neighborhood operation consists of arbitrary Boolean operations (a sum-of-products AND-OR array equivalent) between the set of all the pixels of the data neighborhood, and the set of all corresponding pixels of an arbitrarily specified comparison neighborhood. Any bit of either neighborhood may be independently defined as true, false (complemented), or "don't care". Each of the eight bit-planes forms an independent set of inputs, subject to independent neighborhood operations. As a



result, eight independent one-bit results are obtained from a single pass of the data through the pipeline, yielding an orthogonal eight-bit Boolean vector as output.

Both neighborhood operators are applied independently and simultaneously. They operate on the same data stream, using neighborhood operations which may be different. Their outputs, or the contents of either of the field buffers, may be independently subjected to a second transformation by either of two programmable functions. The first of these is a lookup-table mapping function (TVF L.U.T. in figure 7.) This transformation may be a function of one or two eight-bit arguments. If two arguments are used, they may be taken from homologous pixels of either of the NOP outputs or either of the field buffers. The lookup-table is program-selectable from a number of stored tables which depends on the number of arguments to the input. The other function, with inputs selectable from the same sources, is an ALU with two eight-bit inputs (ALU C.)

Finally, the contents of either of the buffers, the results of either of the neighborhood operators, and the results of either of the two functions of two arguments, may be sent to the three output pathways, or to the four wildcard busses, in any combination, by a crossbar switching network shown at the extreme right in figure 7.

These basic features of the processing stages may be altered in operation by one of two special processing modes. The MIMD, or "region of interest" mode allows each MPS to switch between alternative operation sets on a pixel-by-pixel basis. In this mode, one image buffer of the stage contains a map of the operations to be performed on homologous pixels of the image buffer undergoing operations. In this mode, the contents of the operation-controlling buffer are treated as offsets into the on-board micro-operation store of the stage, and select the operative micro-operation code for each pixel-time. Each micro-operation word controls the routing of data-flow within the stage, the ALU functions, the identity of the look-up tables employed, and the selection of output paths. Potentially, up to 256 different alternative operation sets could be specified by the eight-bit contents of each pixel in the map. In practice, the number of alternative operation sets selectable during a field processing time will be limited by the amount of memory available within the stage to store them, which may be enlarged at will. The operation sets stored in the available memory may be changed arbitrarily between fields.

In "Pyramid Mode", PIPE allows the construction of multiresolution, "pyramid", sequences of images. The basic

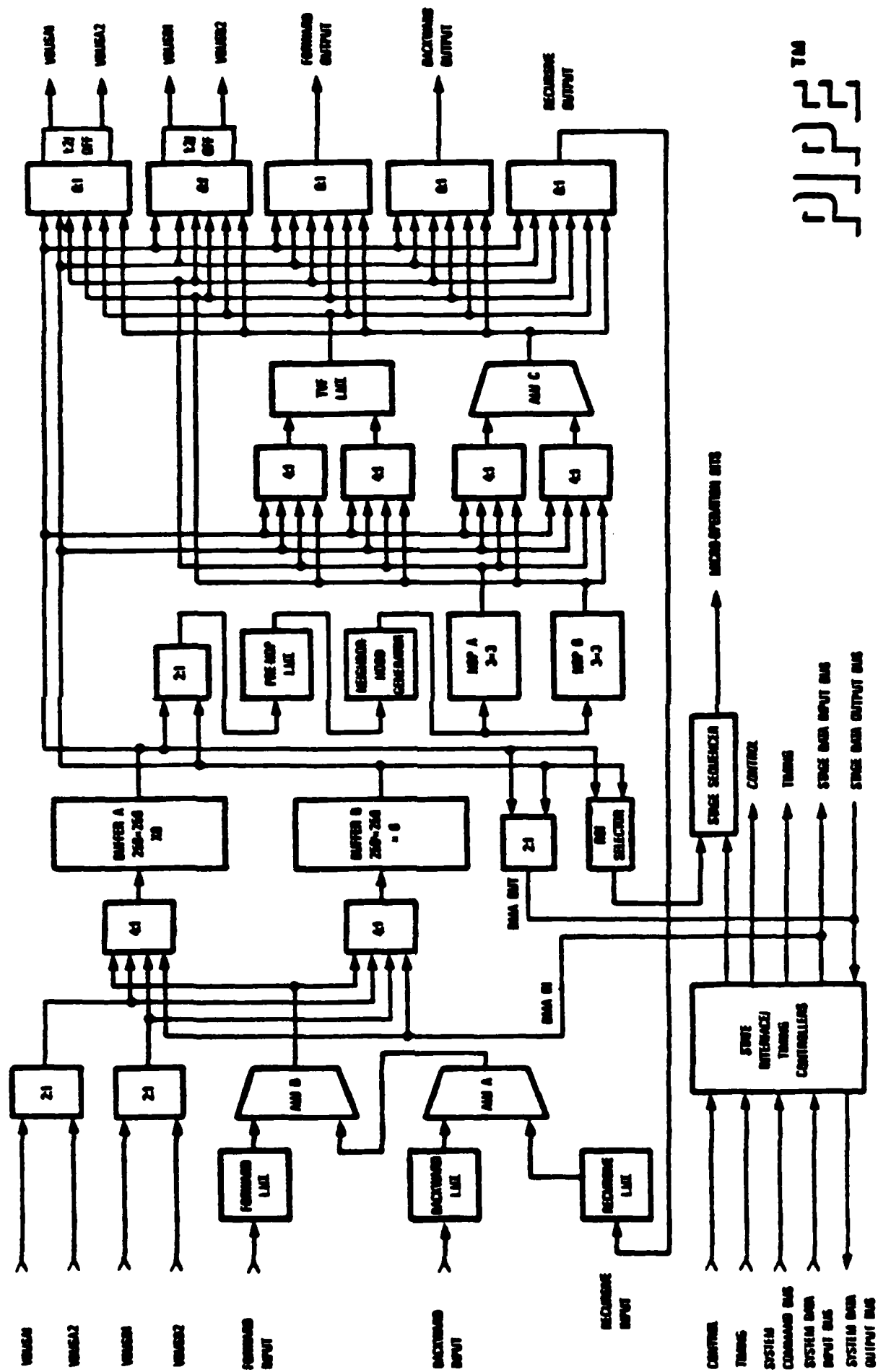


Figure 7. The internal architecture of a PIPE processing stage.

operations available in PIPE for constructing image pyramids are sampling and pixel doubling. Sampling is used to reduce the resolution of an image, while doubling is used to increase the size of an image.

Both the sampling and doubling operations are performed by manipulating addressing rates within a stage. Sampling is achieved via the forward pathway by incrementing the destination image addresses half as fast as the source addresses. That is, on each row, the first pixel in the source image is written to the first pixel in the destination image. The second source pixel is also written to the first destination pixel. The address of the destination pixel is then incremented, and the procedure is repeated. The same process is used to sample into every other row in the destination image. The result is that the destination image is one quarter the resolution of the source image. Doubling is accomplished via the retrograde pathway by the inverse of the sampling process. That is, the addresses in the source image are now being incremented at half the rate of those in the destination image. For each row in the source (reduced-resolution) image, two identical rows are output in the destination image. For each pixel in each row of the source image, two identical pixels are stored in the destination image.

The simple operations of image sampling and pixel doubling are not of themselves very useful except for a narrow range of applications. However, they may be combined with the other operations in the MPS, so that a much broader class of operations becomes possible. Prior to sending the images over the forward and backward paths, they may pass through the operations of the output section of the MPS, and prior to storage, they may pass through the operations of the input section of the destination MPS. For example, the neighborhood operator can be used to smooth the image before sampling. By iterating the neighborhood operation prior to sampling, the effects of neighborhoods larger than three by three can be obtained, allowing, for example, the construction of "Gaussian" pyramids using the hierarchical discrete correlation procedure of Burt<sup>3</sup>. By passing an image through one neighborhood operator into the forward path, through the other into the recursive path, and by passing the retrograde path directly to the look-up table of the input section of the N-1st MPS, the pyramidal neighborhood operations of Tanimoto's Hierarchical Cellular Logic<sup>4</sup> may be realized.

Edge effects that arise when a neighborhood operator is applied are dealt with in the same manner for all resolutions of images, and for borders of MIMD operation regions as well as for frame borders. PIPE automatically provides the replication or zeroing of border pixels. If a neighborhood has a row or a column

that lies outside the boundaries of the image (either beyond the image buffer itself or beyond the extent of a low-resolution image, or beyond the bound of a MIMD operation type), the non-existent pixels are zeroed or replaced by the pixels in the border row or column. For a three by three neighborhood, this is equivalent both to reflecting the image and to repeating the border pixels. This is achieved in the same way as the varying resolution images are constructed, i.e., by manipulating the address lines of the buffer.

### Output Stage

The output stage performs a role at the end of the processing chain similar to that of the input stage at its beginning. The final MPS delivers its forward image output to either one of a pair of field buffers in the output stage, and can simultaneously read from the other buffer of the output stage. The data read from the output stage is used as the input to the retrograde path of the final MPS. Without interrupting the image-processing, either buffer of the output stage can be read from or written into by an external device, which is both the consumer of the processed forward data-flow and the supplier of data for the retrograde path.

### Sequencer and Stage Control

Prior to run time, the pipeline modules must be set up with the individual operations to be carried out. The upper levels of the system must load each stage with appropriate instructions for all the processing steps to be employed. Each stage has storage for up to 256 128-bit micro-operation words, each of which can specify the entire set of operations for the stage. This store is loaded by the host or upper levels of the vision system. Each stage also has control circuitry which allows a micro-operation word to be selected from the stored set by external command. The selection command can come either from the host or from PIPE's Sequencer. In either case the selection can completely reconfigure the stage operation during the inter-field time by selecting a new stage operation set for the next field-time, or a group of stage operation sets from which members are to be selected on a pixel-by-pixel basis in MIMD mode during the next field-time.

Normally, the cycle-by-cycle selection of the operations stored in each stage will be performed by the Sequencer module. When active, this unit issues an operation selection command to every stage of PIPE at the beginning of each field-time. The nature and order of these operation selection commands are determined by a program in the sequencer module which specifies the order of

commands to the stages, including loops and branches, so that the coordination of stage operations to effect various PIPE algorithms is accomplished. This sequencer program is loaded from the host prior to run time. At any time, the host can override the sequencer program and intervene in the stage operation selection process directly. In operation, the upper levels of the system may instruct the sequencer to select a stage program, instruct it to branch in the specified sequence of operations, or permit it to follow the pre-set sequence of operations (which may contain branch points on repetition counts.)

Programming and running PIPE thus consists of specifying the operations to be performed by each stage, loading the corresponding operators, parameters, and tables into the stages, and then loading the sequence of operations for the stages into the sequencer module. For program development, the contents of any buffer and the output of any operator in the system can be displayed on a video monitor, while the sequencer is single-stepped.

#### INPUT/OUTPUT MODES

In keeping with its role as a real-time vision processor, PIPE has several rapid means for transferring information to and from the host device. The wildcard busses may be output directly, either as digital streams or after conversion to analog signals. The analog outputs provide for interface to monitors or other display devices, while the digital outputs provide streaming I/O at video rates to any specialized device which can accept this rate of input. This streaming I/O mode is represented in figure 8-a. Figure 8-b presents a parallel port I/O scheme which allows the host device access to any of the PIPE image buffers in a random access fashion. These parallel port transfers can also function in auto-increment and DMA modes. The most sophisticated method of transferring image information to the host uses the ISMAP device to transfer symbolic descriptions of the image to the host's address space. This transfer may be bi-directional, and can take any PIPE buffer as a source or destination. This I/O technique is illustrated in Figure 8-c.

#### ISMAP

The associated device, ISMAP, will map processed iconic images into symbolic (property-indexed) structures. ISMAP describes the processed images produced by PIPE by mapping the iconic image of Boolean vectors produced by PIPE into ordered list structures in the address space of processors in the upper levels of the system, in real time. A major function of ISMAP is to map PIPE's image addresses into a space ordered by symbolic

picture feature descriptors. This saves processors in the upper levels of the system the necessity of scanning the processed image to find locations of items of interest. Physically, ISMAP resides in the PIPE backplane as an integrated part of the system.

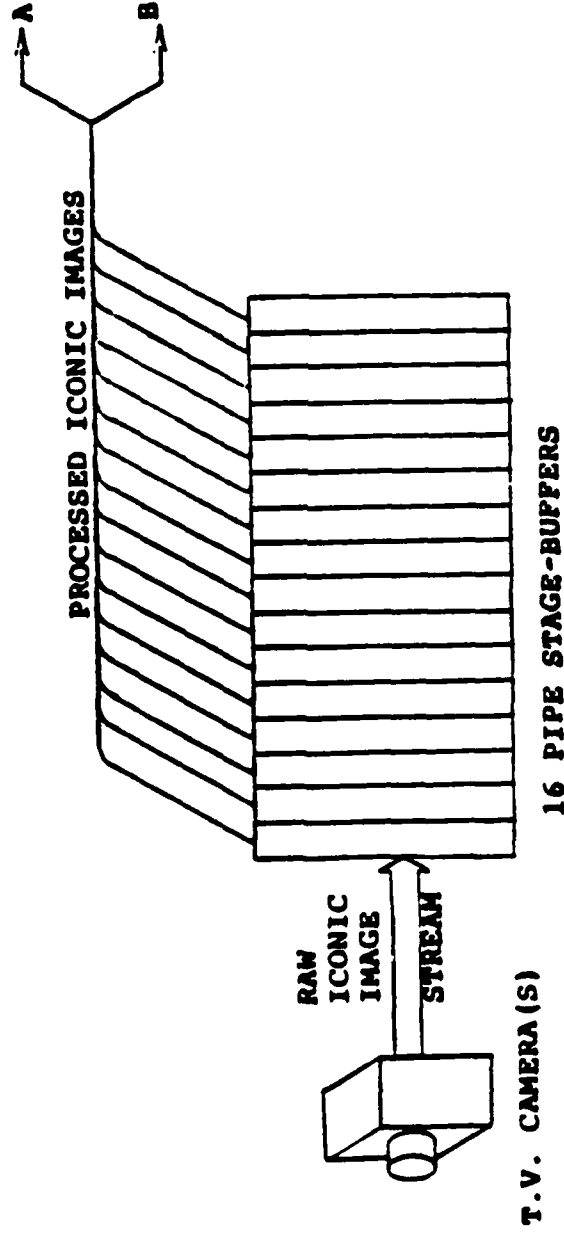
In each frame time, ISMAP scans the entire image of iconically-ordered feature codes produced by PIPE, and prepares three organized descriptions of the feature set of the image in a symbolically-ordered space in the address space of the host. The three descriptions of the image's feature content produced by ISMAP are: 1) a histogram of feature types, organized by feature type, 2) a cumulative histogram of feature types organized by feature type, and 3) lists, organized by feature type, of the image-coordinates of every feature in the image. The values in the cumulative histogram serve as pointers to the heads of the lists of image locations. For example, if the host needs the locations of all edges with a certain angle of inclination, it can find the beginning and the length of a compact list of those locations, in its own address space, simply by looking at the cumulative histogram entry for the desired edge direction. It may first use the histogram to determine what edge directions are prominent in the image, or perform any of a variety of other algorithms on this information without the burden of examining the image-space in order to count or locate features of interest.

ISMAP may also run in reverse, creating images in host buffers from feature lists created or selected by the host. This permits ISMAP to be the first level of the upper system which can generate hypothesis images for PIPE, which allows the PIPE/ISMAP combination to perform operations such as Hough transforms easily.

#### CONMAP

ISMAP is one of two PIPE auxiliary devices. Another, CONMAP, which is currently under construction, stands at the front of PIPE as a geometric pre-processor for input images. The CONMAP device accepts iconic images and remaps them into topologically identical, but geometrically different iconic images prior to their acquisition by PIPE. In its final form, CONMAP will implement fully general conformal mappings on images. There are a variety of uses for such transforms. Log-polar mappings have been suggested by Weiman and Chaikin<sup>5</sup>, as a means of converting complex image properties, such as rotation and scale, into simple translations. Jain<sup>6</sup> has employed a similar transformation for utilizing the effects of camera motion in image understanding. Other applications include simulation of lens geometries which permit non-uniform resolution (e.g. a high-

2 CHANNELS OF  
PIXEL-RATE OUTPUT IN  
RASTER-SCAN ORDER  
(DIGITAL OR RS-170 ANALOG)

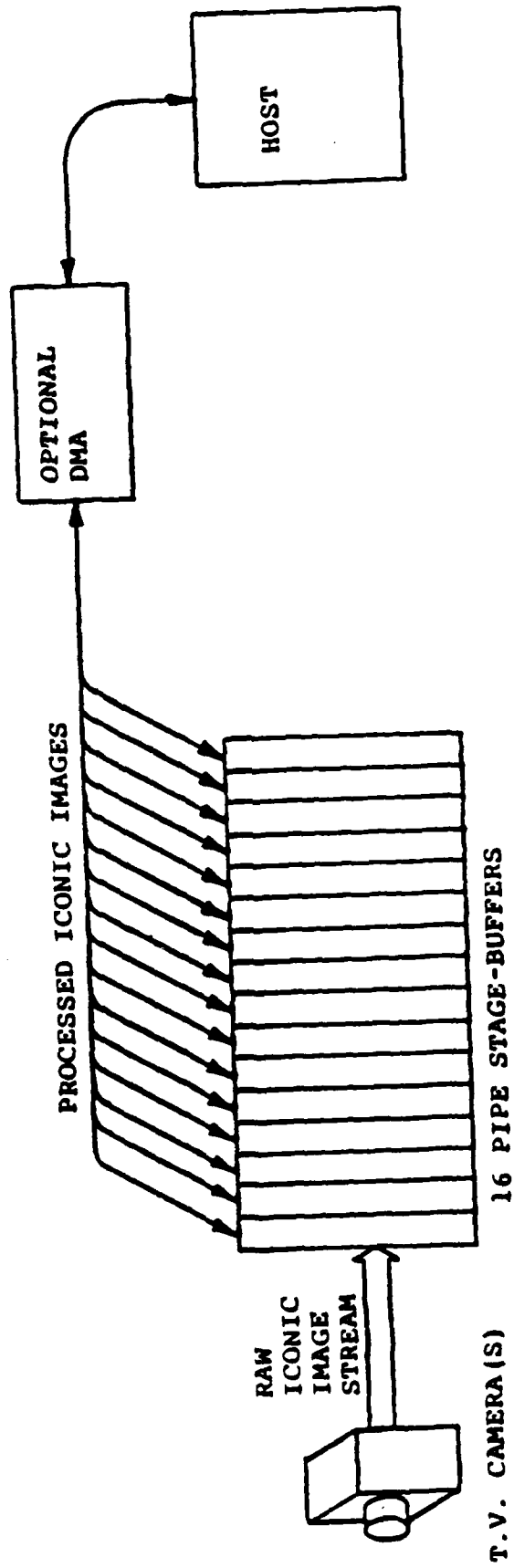


# STREAMING I/O

Figure 8. The I/O modes supported by PIPE. a.) streaming pixel I/O, b.) parallel port random access, c.) bi-directional ISMAP operation. (8 b and 8-c on next pages)

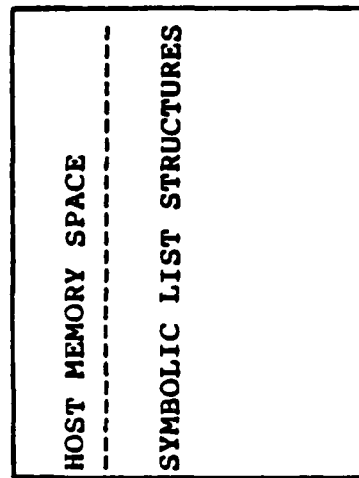
**BI-DIRECTIONAL PARALLEL PORT TRANSFERS:**

- RANDOM ACCESS
- AUTO-INCREMENT
- DMA

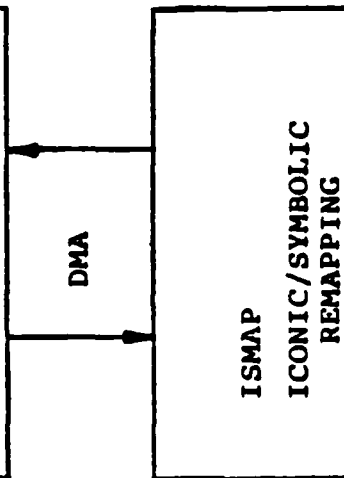


PARALLEL PORT I/O

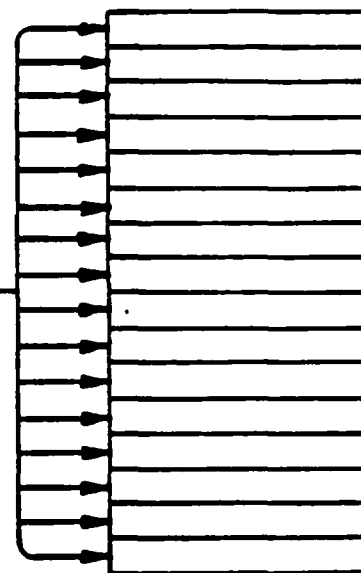




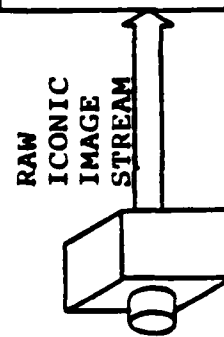
BI-DIRECTIONAL STREAM OF  
PROCESSED SYMBOLIC IMAGE  
DESCRIPTIONS



BI-DIRECTIONAL STREAM OF  
PROCESSED ICONIC IMAGES



16 PIPE STAGE-BUFFERS



ISMAP I/O

resolution "fovea" with lower-resolution periphery.) CONMAP will transform images at field rate, and its operation will be transparent to PIPE except for a one-field delay.

#### PIPE's SYSTEM SOFTWARE.

PIPE is provided with a software toolkit which includes a number of highly-developed programming systems. A simulator exists, although it is not used when a PIPE machine is available, since the other software development tools permit interactive test and debugging of algorithms in real-time on the machine. All PIPE system software is graphics-based and runs in color-coded formats on the screens of the host.

At the bottom level, a software "keyboard" permits interactively displaying and setting any bit, byte, or word in the machine's microcoded operation stores. This can be done while PIPE is running algorithms, and the effects observed. This program can also save code to, or load code from disk files.

Microcoded instruction lists for the operation store can be produced and stored in disk files by a graphics-based assembler. This program presents diagrammatic icons portraying the logical functions of the processing stages on the screen, and interactively fills in data-flow paths and operations as the user specifies them. The program has an expert-system knowledge of PIPE. It will only inquire about options as they become necessary based on user choices, and it will reject contradictory or illegal choices.

Figure 9. demonstrates a PIPE programming diagram. Spatial progression of stages is from left to right. Temporal progression of machine cycles is from top to bottom. A single row indicates the state of the machine in a single field time; a single column indicates the successive states of a single stage. Diagrams such as this illustrate the space-time systolic nature of PIPE programs, and can be transferred directly to the graphics screens of the system software tools for program generation.

Another graphics-oriented program, similar to an assembler, generates programs for the global sequencer which, in turn, controls selection of words in the micro-operation stores of all PIPE's stages on a cycle-by-cycle basis. This program generates disk files which can be loaded into PIPE's system controller.

PIPE employs many tables of great complexity. A table-generating program is available to simplify their production. For example, extremely complex tables are required for the bit-slice arithmetic in the convolvers. The table entries have no simple

relationship to the weights of the neighborhood mask. The table-generating program will present a graphic representation of the neighborhood, query the user for mask values, and prepare the required tables and save them in disk files for later loading into PIPE. A variety of other table types are also handled. A related program finds the optimal sequence of  $3 \times 3$  kernels for the production of any  $N \times N$  convolution.

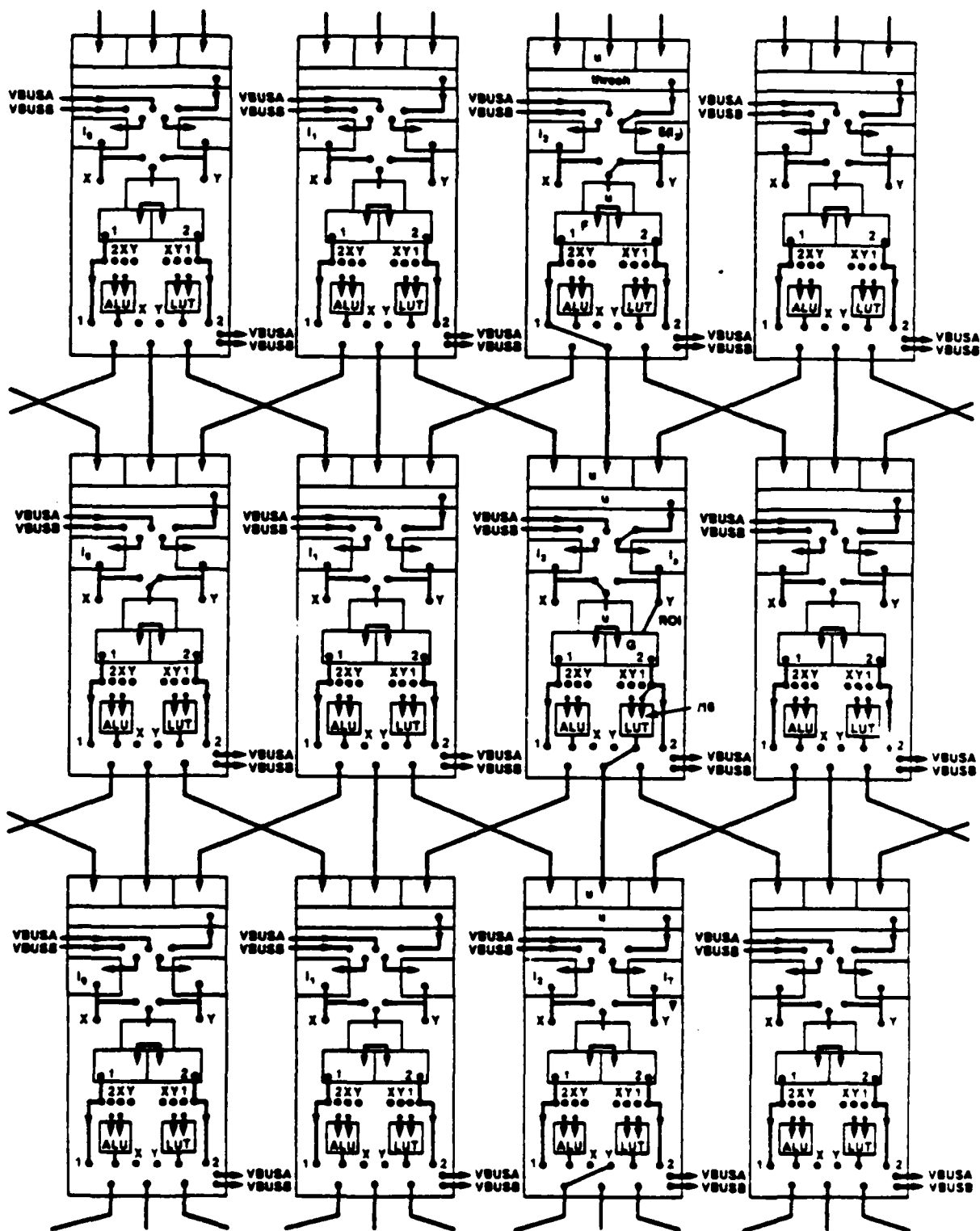
At the next level, PIPE is programmable in PIPE Command Language (PCL). In PCL, it is possible to refer to stages, storage areas, and operational units of the machine, to command their states, and to load them from buffers in the host.

A graphics-oriented interactive monitor program permits real-time manipulation of PIPE with PCL. This functions much like a low-level monitor program on a traditional computer, except that the command set is the full set of a powerful macro-assembly language for the machine. The monitor is useful principally in debugging programs that have been generated by other means, but it is possible to build short test programs with it.

The principal means of generating PCL programs is by means of a low-level compiler which accepts commands in PIPE Intermediate Format (PIF). PIF is a higher-level language which permits the programmer to command successive states of machine operations, to refer to named disk files, and to associate the files with the machine entities. Such files may be the micro-operation lists, sequencer programs, and tables generated by auxiliary programs. These files represent operations of broad general utility, and form a library upon which PIF programmers can draw.

This compiler can operate either from PIF programs stored in disk files, or interactively with the user as he issues PIF commands. Most of the existing applications software for PIPE has been written in PIF. It is intended as the intermediate format which will be generated by compilers for higher-level image-processing languages now under construction.

Software tools already available for PIPE would permit a competent PIPE programmer to write all PIPE software required for most applications at a realistic level and with reasonable efficiency, although the task would be made easier if higher-level software tools currently being produced were available now. A growing body of basic application routines, for feature detection, image enhancement, and similar elementary functions already exists. About one day's time is sufficient for a



**Figure 9.** A PIPE programming diagram. Spatial progression of stages is from left to right. Temporal progression of machine cycles is from top to bottom. A single row indicates the state of the machine in a single field time; a single column indicates the successive states of a single stage.

competent programmer to write such a routine. Experience suggests that a doctoral-level computer scientist can become an expert PIPE programmer in less than one month's time.

#### DISCUSSION.

PIPE is designed as a front-end processor for low-level iconic-to-iconic image processing. It is intended to perform transformations on images to extract features similar to those in the primal sketch of Marr.<sup>7</sup> These features make intensity changes and local geometric relations explicit in images, while maintaining the spatial representation. In this, PIPE differs from many processors designed for image-processing. These other processors are usually designed to perform both local and global image-processing tasks, often in an interactive environment.

A recent survey by Reeves<sup>8</sup> divides image-processing tasks into two classes. Low level image processing usually modifies parts of images, but maintains the image array. Higher level processing, however, works on symbolic representations of the contents of images. Low level processing has usually given rise to architectures based on single instruction stream, multiple data stream (SIMD) structures. The higher level functions are usually carried out using processors based on multiple instruction stream, multiple data stream (MIMD) structures. The design of PIPE allows it to act as a SIMD pipeline, or as a (restricted) MIMD pipeline. The MIMD mode is entered whenever the region-of-interest operators are used. The limitations on these operators are that there are at most 256 different operators available per stage, and that using the region of interest generally precludes using some other operators, such as the functions of two arguments. Using the retrograde pathway to insert expectations from the host into the image analysis process also blurs the distinction between high level and low level processes.

Some general features of PIPE's architecture, multiple pipelined planes of processing, and the concept of the iconic-to-symbolic remapping performed by ISMAP, are drawn in spirit from a more elaborate machine proposed but never constructed by McCormick, Kent, and Dyer.<sup>1,2</sup> That device was in turn inspired by certain observations on the nature of processing in the visual cortex. Although PIPE is in some respects a simpler device, it also carries that analogy further with the implementation of the backward pathway which emulates the connectivity in the biological vision system, where, with the exception of the final link between the retina and the lateral geniculate nucleus, there are in fact more fibers descending from higher to lower

levels of processing than vice versa. The arrangement of interactions among the three pathways, with combination of images into stage buffers and separate operations available into emerging pathways also is drawn from this model.

#### CONCLUSIONS.

This paper has described a new image pre-processor, consisting of a sequence of identical stages, each of which can perform a number of point and neighborhood operations. An important feature of the processor is the provision of forward, recursive, and backward paths to allow image data to participate in temporal as well as spatial neighborhood operations. The backward pathway also allows expectations or image models to be inserted into the system by the host, and to participate in the processing in the same way as images acquired from the input device. The region-of-interest operator is also a powerful, and unique, feature of PIPE, allowing the results of feature-extraction processes to guide further image analysis. PIPE also provides a multi-resolution capability, enabling global events to be made local. This is important in a machine that has only local operators. Much research needs to be done to explore the capabilities of the device but early experiments indicate that the system will have a wide range of applications in low-level real-time image processing.

## References

1. B. McCormick, E. W. Kent, and C. Dyer, A Visual Analyzer for Real-Time Interpretation of Time-Varying Imagery., in: "Multicomputers and Image Processing 3.", K. Preston and L. Uhr (Eds.), Academic Press, 1982.
2. B. McCormick, E. W. Kent, and C. Dyer, A Cognitive Architecture for Computer Vision. in: "Fifth Generation Computer Systems.", T. Moto-Oka (Ed.), North Holland, 1982.
3. P. J. Burt, Fast hierarchical correlations with Gaussian-like kernels. Proc. Fifth International Joint Conference on Pattern Recognition, Miami, Florida, 1980.
4. S. L. Tanimoto, A Hierarchical Cellular Logic for Pyramid Computers. J. Parallel and Distributed Computing, 1, 105-132, 1984.
5. C. Weiman, and G. Chaikin, Logarithmic Spiral Grids for Image Processing and Display., Computer Graphics and Image Processing, 11, 197-226, 1979.
6. R. Jain, Segmentation of Frame Sequence Obtained by a Moving Observer., General Motors Research Publications: GMR-4247, 1983.
7. D. Marr, Early processing of visual information. Phil. Trans. Royal Society B.275, 1976.
8. A. P. Reeves, Parallel computer architectures for image processing. Computer Vision, Graphics, and Image Processing 25, 1984, 68-88.

## *A new method for characterization of shape*

O. SKLIAR\* and M.H. LOEW

*Department of Electrical Engineering and Computer Science, George Washington University, Washington, DC 20052, USA*

Received 28 December 1984

Revised 29 May 1985

**Abstract:** A method is presented that uses a diffusion-like process to describe the shape of a region. Convexity is not required, the descriptor is invariant under several common transformations, is applicable in the  $n$ -dimensional case, and is easy to compute.

**Key words:** Shape characterization, image-processing, diffusion process, pattern recognition.

### 1. Introduction

Shape description is an essential component of any image-understanding system. Many approaches to description of shape have been proposed and used in the fields of image processing and computer vision. Pavlidis (1978) suggested a taxonomy of shape descriptors based on: (1) whether just the boundary, or the entire interior of the object was examined (the techniques were called external and internal, respectively); (2) whether the characterization was made on the basis of a scalar transform (in which a picture is transformed into an array of scalar features), or a space transform (a picture is transformed into another picture); and (3) whether the procedure is or is not information-preserving in the sense that the original image can be reconstructed from the shape descriptors.

Existing methods include the  $\psi-s$  curve, in which  $\psi$  is computed as the angle made between a fixed line and a tangent to the boundary of the region; it is plotted against  $s$ , the arc length of the boundary traversed. For a closed boundary, the function is periodic, and may be associated with segmentation of the boundary in terms of straight

lines and circular arcs (Ballard and Brown, 1982). Other methods evaluate *eccentricity* (or *elongatedness*) in a variety of ways, including length-to-width ratio and ratio of the principal axes of inertia; *compactness* (e.g.,  $\text{perimeter}^2/\text{area}$ , and Danielsson's method (Danielsson, 1979)); the *slope-density function*, which is a histogram of  $\psi$  collected over the boundary; *curvature*, the derivative of  $\psi$  as a function of  $s$ ; projections of the figure onto an axis (the *signatures*); *concavity* with a tree of regions that will create the convex hull of the original object; *shape numbers* based on chain-coding of the boundary; and the *medial-axis transform*, which transforms the original object to a stick figure that approximates the skeleton of the figure.

We present here a new shape measure that allows rapid assignment of labels that are both intuitively appealing and rigorously based. The descriptor can be computed easily on existing hardware and may be implemented immediately on future parallel-processing systems. Regions need not be convex (although the modest requirement is imposed that each region be simply-connected; i.e., have a single inside and a single outside). This is a significant advantage in light of the comment by Pavlidis (1978) that there exist a number of shape description techniques applicable *only* to convex objects, while

\* Visiting from the Department of Physics, National University of Costa Rica.



some of the general ones perform much better if restricted to that class. The method described below works equally well for convex and non-convex regions. Further, the approach can be extended immediately to three-dimensional objects.

This is the first in a series of papers examining the behavior of a diffusion-type shape descriptor. With respect to the taxonomy noted above, it is internal, scalar, and non-information preserving.

## 2. Method

The diffusion-type procedure simulates the release at an initial time of a given number of particles from each pixel along the boundary of a region to be studied. At each instant of discrete time thereafter, new values of pixel contents are computed based on an assumed diffusion constant and the isotropic assumption (i.e., that the diffusion law applies equally in all directions for all parts of the region under study). The process consists of an initial transient and a subsequent steady-state condition. In steady-state all pixels contain the same number of particles. During the transient, however, the number of particles in each boundary pixel depends upon the shape of the boundary. The concentration is greater in concavities than in convexities, with straight or nearly-straight regions having intermediate concentrations. It is necessary therefore to stop the diffusion process during the transient to detect these characteristics of the boundary. When the simulated diffusion process is stopped, the sequence of numbers of particles in the boundary pixels can be used to generate a shape-related code.

This approach is implemented easily on digital computers so that the effects of changes in the following relevant process parameters can be studied: constant of diffusion, stopping-time, and initial number of particles per pixel.

Let  $N_{i,j}(t)$  be the number of particles contained in the pixel at coordinates  $(i,j)$  at time  $t$ . Then the fundamental algorithm to be utilized is:

$$\begin{aligned} N_{i,j}(t+1) = & N_{i,j}(t) - 4KN_{i,j}(t) \\ & + K(N_{i-1,j}(t) + N_{i+1,j}(t) \\ & + N_{i,j-1}(t) + N_{i,j+1}(t)). \end{aligned} \quad (1)$$

Equation (1) expresses the requirement that the number of particles in a given pixel of the image at time  $t+1$  equals the number of particles that were there at  $t$ , minus the number of particles that were transferred by the assumed diffusion process to the (4-)neighboring pixels, plus the number of incoming particles from those same neighbors, based on their respective contents at  $t$ . Neighbors that lie outside the region do not participate in the process of equation (1).

Though in this preliminary communication we will consider only the two-dimensional case, the approach can be generalized easily to any number of dimensions. In the three-dimensional case the basic algorithmic equation is:

$$\begin{aligned} N_{i,j,k}(t+1) = & N_{i,j,k}(t) - 6KN_{i,j,k}(t) \\ & + K(N_{i-1,j,k}(t) + N_{i+1,j,k}(t) \\ & + N_{i,j-1,k}(t) + N_{i,j+1,k}(t) \\ & + N_{i,j,k-1}(t) + N_{i,j,k+1}(t)). \end{aligned} \quad (2)$$

In general, for the  $n$ -dimensional case, if we call  $N_{x_1, x_2, \dots, x_n}(t)$  the number of particles contained in the pixel  $x$  at coordinates  $x_1, x_2, \dots, x_n$  at time  $t$ , then the following equation will apply:

$$\begin{aligned} N_{x_1, x_2, \dots, x_n}(t+1) = & N_{x_1, x_2, \dots, x_n}(t) - 2nKN_{x_1, x_2, \dots, x_n}(t) \\ & + K(N_{x_1-1, x_2, \dots, x_n}(t) + N_{x_1+1, x_2, \dots, x_n}(t) \\ & + N_{x_1, x_2-1, \dots, x_n}(t) + N_{x_1, x_2+1, \dots, x_n}(t) \\ & + \dots + N_{x_1, x_2, \dots, x_n-1}(t) + N_{x_1, x_2, \dots, x_n+1}(t)). \end{aligned}$$

For the problem that we are considering we do not need to adjust the parameter  $K$  to experimental data - as should be done in the case of simulation of a real diffusion process of matter or heat. So, for purposes of computation we can assign to  $K$  any value in the range  $0 < K < 1$ . The smaller  $K$  is, the higher will be the degree of detail of the results of the diffusion-type process, but of course at the expense of additional computer time. The tradeoff between detail and accuracy on one hand and computer time on the other will be discussed critically in the next paper of this series.

### 3. Preliminary results

The algorithm was tested with two shapes: a square, and an irregular region. The corresponding results are presented. At the initial time ( $t=0$ ), 10000 particles were assigned to each boundary pixel for each of the two cases. In the case of the square, the number of particles for each pixel of the image has been computed for times 10, 50, and 100. (See Figures 1a, b, and c, respectively.) For

|      |      |      |      |      |
|------|------|------|------|------|
| 9923 | 9158 | 9090 | 9158 | 9923 |
| 9158 | 1682 | 949  | 1682 | 9158 |
| 9090 | 949  | 157  | 949  | 9090 |
| 9158 | 1682 | 949  | 1682 | 9158 |
| 9923 | 9158 | 9090 | 9158 | 9923 |

Figure 1(a). Number of particles in each pixel for the square region, with  $k=0.01$ . (a)  $t=10$ .

|      |      |      |      |      |
|------|------|------|------|------|
| 8940 | 7605 | 7106 | 7605 | 8940 |
| 7605 | 4654 | 3541 | 4654 | 7605 |
| 7106 | 3541 | 2197 | 3541 | 7106 |
| 7605 | 4654 | 3541 | 4654 | 7605 |
| 8940 | 7605 | 7106 | 7605 | 8940 |

Figure 1(b).  $t=50$ .

the three cases a value of  $K=0.01$  was used.

Let us assume that we establish an order on the boundary of those squares by labeling the pixels with consecutive natural numbers following a clockwise direction, starting at the left uppermost. In Figures 2a, b and c the number of particles on the boundary pixels has been expressed as a function of the number utilized as labels for the pixels, for each of the corresponding cases of Figure 1.

In the case of the irregular shape, only the number of particles of each pixel on the boundary has been shown for times 3 and 10 in Figures 3a and b.

For such irregular shapes the boundary pixels are labeled with consecutive integers, again proceeding clockwise from the left uppermost pixel. Graphics of the same kind as Figure 2 appear as Figure 4 for the irregular shape.

Both for the case of the regular shape (the square) and the irregular one, it can be seen immediately that the number of particles per pixel - the concentration - is higher in concavities than in convexities.

If human shape perception does rely heavily on detection of curvature maxima (Attneave, 1954), then the (positive- and negative-going) peaks in a plot of pixel content-vs.-boundary location (corresponding, respectively, to segments of high con-

|      |      |      |      |      |
|------|------|------|------|------|
| 6406 | 6402 | 6399 | 6402 | 6406 |
| 6402 | 6398 | 6395 | 6398 | 6402 |
| 6399 | 6395 | 6393 | 6395 | 6399 |
| 6402 | 6398 | 6395 | 6398 | 6402 |
| 6406 | 6402 | 6399 | 6402 | 6406 |

Figure 1(c).  $t=500$ .

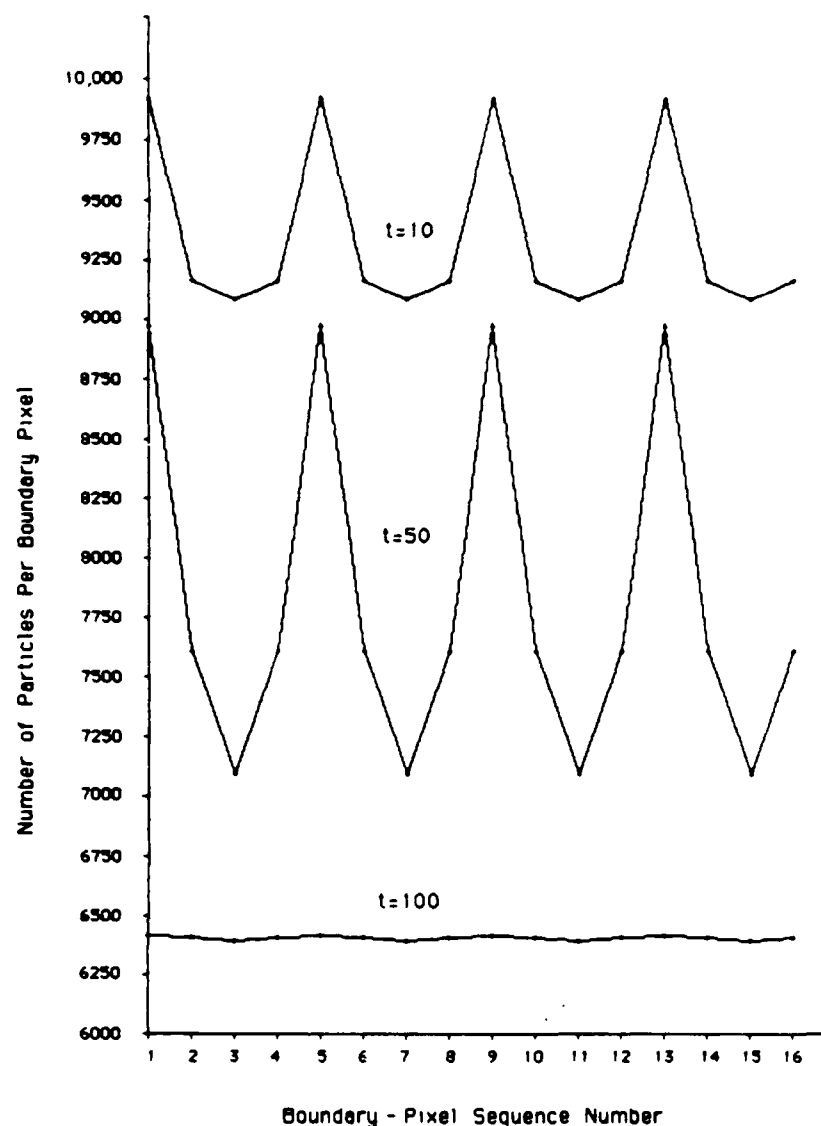


Figure 2. Number of particles for consecutive boundary pixels of the three cases of Figure 3, beginning at the left uppermost (a)  $t = 10$ ; (b)  $t = 50$ ; (c)  $t = 500$ .

cavity and convexity) are likely to be useful in identifying those important regions.

#### 4. Conclusions and perspectives

We have presented a new method for describing the shape of a region. The region must be simply-connected, but need not be convex. The descriptor is invariant under translation, rotations by

multiples of  $90^\circ$ , and, it appears, under scale changes. It is almost invariant – in a sense that will be made precise in the next paper – under rotations of non-multiples of  $90^\circ$ . It can be implemented easily in hardware (especially on future parallel processors), is as effective in higher dimensions as in two, and will lend itself easily to image-processing and pattern-recognition applications. The method appears to be relatively insensitive to noise (cf. the medial-axis transform, in which very

|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
|      | 7780 | 7570 | 7550 | 7550 | 7550 | 7550 | 7550 | 7550 | 7550 | 7550 | 7550 | 7550 | 7550 | 7580 | 8000 | 9250 |      |
| 7780 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 8000 |      |
| 7570 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 7600 |      |
| 7550 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 7780 |      |
| 7760 |      |      |      |      |      |      |      |      |      |      |      |      | 5990 | 7360 | 7770 |      |      |
|      | 6020 |      |      |      |      |      |      |      |      |      |      | 4450 |      |      |      |      |      |
|      |      | 5990 |      |      |      |      |      |      |      |      |      |      | 5990 |      |      |      |      |
|      |      |      | 4450 |      |      |      |      |      |      |      |      |      |      | 6020 | 7540 |      |      |
|      | 7560 | 6010 |      |      |      |      |      |      |      |      |      |      |      |      |      | 6050 |      |
| 7780 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 7780 |
| 7600 |      |      |      |      |      |      |      | 6010 | 7130 | 6010 |      |      |      |      |      |      | 7800 |
| 8000 |      |      |      |      | 6020 | 7360 | 7750 |      |      |      |      | 7760 | 7570 | 7550 | 7570 | 7790 |      |
| 9250 | 8000 | 7580 | 7570 | 7760 |      |      |      |      |      |      |      |      |      |      |      |      |      |

Figure 3(a). Number of particles in each boundary pixel for the irregular region (interior pixels' values omitted for clarity), with  $k = 0.01$ ,  $t = 3$ .

large changes in the axis are produced by very small changes in the boundary (Nevatia, 1982)), does not pose problems with the definition of slope (Rosenfeld and Johnston, 1973) as occurs in the  $\psi-s$  curve computation, and appears to be capable of dealing with the matching of partially-occluded shapes (e.g., in the robot vision case), since the diffusion-produced boundary descriptors are likely to be less affected far from the occluding boundary and thus can provide the basis for a partial match to a pre-stored description of a complete boundary. The effects of noise and of occlusion will be studied together in a forthcoming paper.

The effect of changes in the diffusion constant,  $K$ , and the time at which the process is stopped will be examined in detail in future papers. A method and its proof are needed that will allow for in-

variance of the measure under scale changes; these results are expected soon. The present stopping criterion, which terminates the process when the difference between maximum and minimum values along the boundary is maximized, has great intuitive appeal, since we are interested in distinguishing concavities from convexities and can think of this as a sensitivity measure.

Alternatives for the characterization of the results of the diffusion-type process also will be considered in the future. Particle-count plotted against boundary position is not necessarily the most effective representation. Other possibilities are: (1) to normalize the count by subtracting the mean and dividing by the standard deviation, thus providing for the possibility of establishing data-independent criteria for detecting extrema; and (2)

|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
|      | 6006 | 5235 | 5116 | 5114 | 5113 | 5114 | 5113 | 5114 | 5116 | 5126 | 5171 | 5326 | 5782 | 6711 | 7682 |      |
| 6019 |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 6780 |      |
| 5610 |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 6130 |      |
| 5494 |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 6148 |      |
| 5660 |      |      |      |      |      |      |      |      |      |      |      | 3775 | 4982 | 5849 |      |      |
|      | 4099 |      |      |      |      |      |      |      |      |      | 2406 |      |      |      |      |      |
|      |      | 3571 |      |      |      |      |      |      |      |      |      | 3550 |      |      |      |      |
|      |      |      | 2423 |      |      |      |      |      |      |      |      |      | 4045 | 4996 |      |      |
|      | 5284 | 4039 |      |      |      |      |      |      |      |      |      |      |      |      | 4664 |      |
| 6083 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      | 6156 |
| 6100 |      |      |      |      |      |      |      |      | 3863 | 4083 | 3898 |      |      |      |      | 6333 |
| 6776 |      |      |      |      | 4138 | 4843 | 5410 |      |      |      |      | 5619 | 5421 | 5395 | 5684 | 6167 |
| 7684 | 6721 | 5843 | 5549 | 5652 |      |      |      |      |      |      |      |      |      |      |      |      |

Figure 3(b). Number of particles in each boundary for the irregular region (interior pixels' values omitted for clarity), with  $k = 0.01$ ,  $t = 10$ .

to use the first difference along the boundary to detect regions of small and of large change.

#### Acknowledgements

We appreciate the computer simulation work of Mr. Jean-Fernand Duc, the figures drawn by Ms. Nevine El-leithy, and financial support provided by the U.S. Office of Naval Research under Contract No. N00014-83-K-0578 and by the Center for Devices and Radiological Health, U.S. Food and Drug Administration under Purchase Order FDA-204358-01-4-F100.

#### References

- Attneave, F. (1954). Some informational aspects of visual perception. *Psychol. Rev.* 61, 183-193.
- Ballard, D. and C. Brown (1982). *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ.
- Danielsson, P.-E. (1978). A new shape factor. *Computer Graphics and Image Processing* 7, 292-299.
- Nevatia, R. (1982). *Machine Perception*. Prentice-Hall, Englewood Cliffs, NJ.
- Pavlidis, T. (1978). A review of algorithms for shape analysis. *Computer Graphics and Image Processing* 7, 243-258.
- Rosenfeld, A. and E. Johnston (1973). Angle detection on digital curves. *IEEE Trans. Comput.* 22, 875-878.

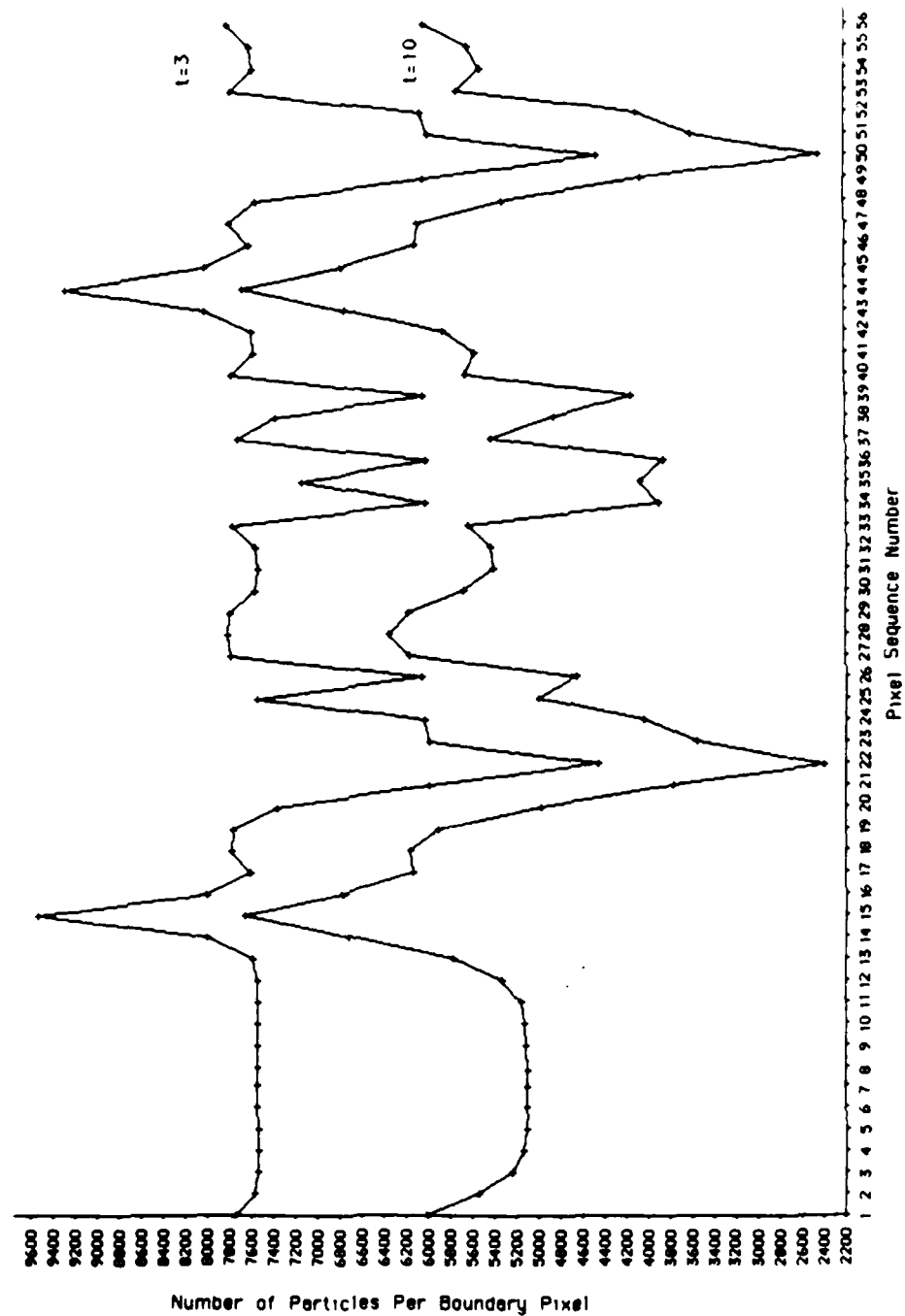


Fig. 4. Number of particles for consecutive boundary pixels of the two cases of Figure 5, beginning at the left uppermost. (a)  $t = 3$ .  
(b)  $t = 10$ .

# THE QUADCODE AND ITS ARITHMETIC

Shu-Xiang Li\* and Murray H. Loew\*\*

\*Dept. of Automatic Control, Changsha Institute of Technology  
Changsha City, Hunan Province, China

\*\*Dept. of Electrical Engineering and Computer Science  
George Washington University, Washington, D.C. 20052, USA

## INTRODUCTION

The quadcode is a hierarchical data structure for describing digital images. It has the following properties:

- Straightforward representation of dimension, size, and the relationship between an image and its subsets.
- Explicit description of geometric properties, such as location, distance, and adjacency.
- Ease of conversion from and to raster representation.

The quadcode has applications to computer graphics and image processing because of its ability to focus on selected subsets of the data and to allow utilization of multiple resolutions in different parts of the image. A related approach is the quadtree. Samet recently presented a thorough survey of the literature in that field [1]. Gargantini [2] and Abel [3] presented linear quadtrees and linear locational keys which are efficient labeling techniques for quadtrees. In those papers, the geometric concepts of the image are discussed by using the tree as an

interpretive medium, and the approaches and procedures are based on traversal of the nodes in the tree. In this paper we present the quadcode system which is a direct description of the image, and discuss the geometric concepts in terms of the coded images.

## 1. QUADCODE

### 1.1 QUADCODE

The quadcode is a quaternary (base 4) code. A quadcode of length  $n$  is of the form

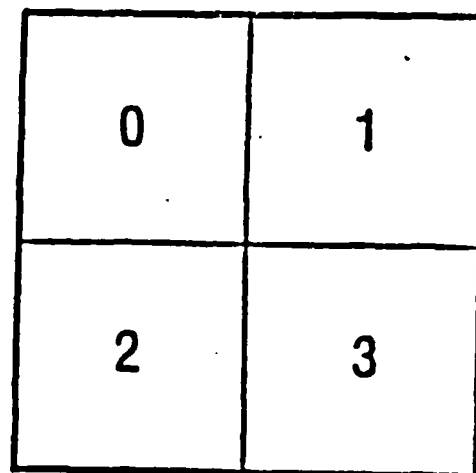
$$Q = q_1 q_2 \dots q_n$$

where  $q_i = 0, 1, 2, 3$  for  $i = 1, 2, \dots, n$ .

When the quadcode is used in describing an image, each character represents one operation of subdividing the image or its subimage into quadrants, as shown and labeled in Fig. 1.

Fig. 1





|   |   |
|---|---|
| 0 | 1 |
| 2 | 3 |

Fig. 1 Quadcodes and Quadrants

In many applications (e.g., smoothing, edge detection, shape description) an image is usually subdivided into much smaller units, so the subdividing operation can be repeated recursively many times, until there is no further subdividing needed. Fig. 2 shows the subdividing process and the corresponding quadcodes. A particular quadrant is represented by one of 0, 1, 2, or 3, concatenated to the quadcode of its predecessor, and after each subdividing operation the length of the quadcode increases by one. As shown in the figure, the quadcode length signifies how many subdividing operations have been done.

Fig. 2

## 1.2 PROPERTIES OF THE QUADCODE

### Property 1 (qc length)

The quadcode length of individual pixels in a  $2^n$  by  $2^n$  image is  $n$ .

### Property 2 (dimension)

The side length of a subimage with quadcode length  $m$  in a  $2^n$  by  $2^n$  image is  $2^{n-m}$ .

|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |
| 2 | 3 | 2 | 3 |   |
| 0 | 1 | 0 | 1 |   |
| 2 | 3 | 2 | 3 |   |
| 2 |   |   |   | 3 |
| 0 | 1 | 2 |   |   |
| 2 | 3 | 3 |   |   |

(a)

|     |     |     |     |   |
|-----|-----|-----|-----|---|
| 000 | 001 | 010 | 011 | 1 |
| 002 | 003 | 012 | 013 |   |
| 020 | 021 | 030 | 031 |   |
| 022 | 023 | 032 | 033 |   |
| 20  |     | 21  |     | 3 |
| 22  |     | 23  |     |   |

(b)

Fig. 2 Subdividing and Quadcode Representation

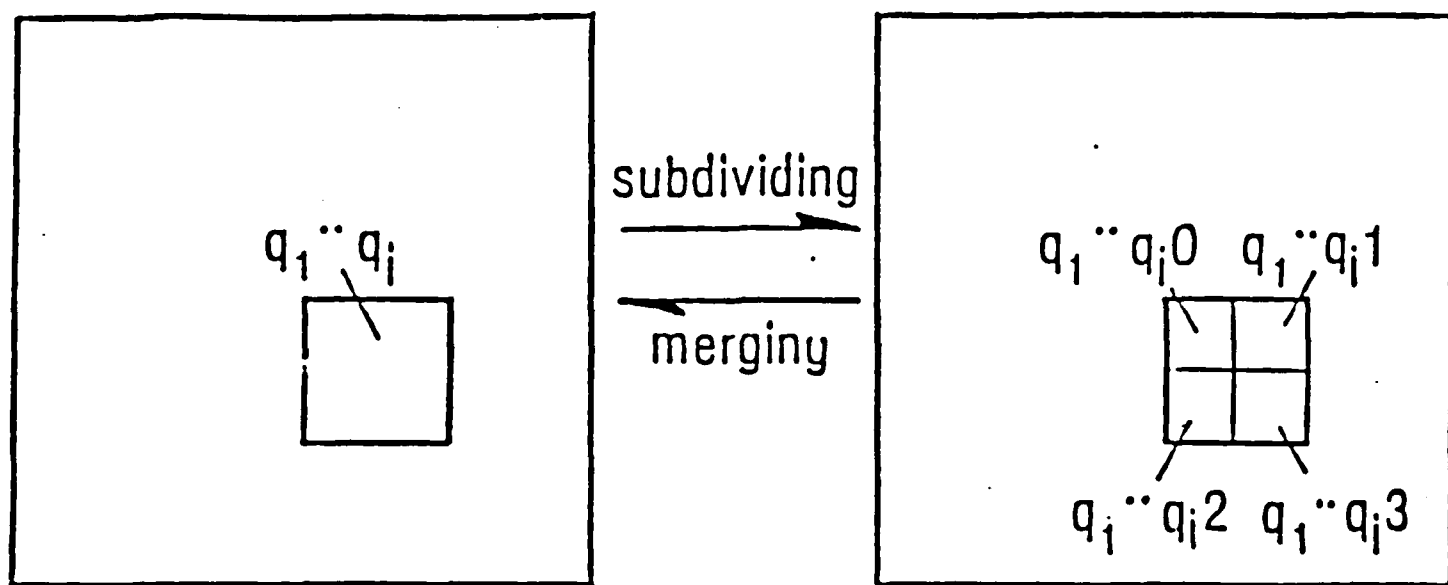


Fig. 3 Subdividing and Merging

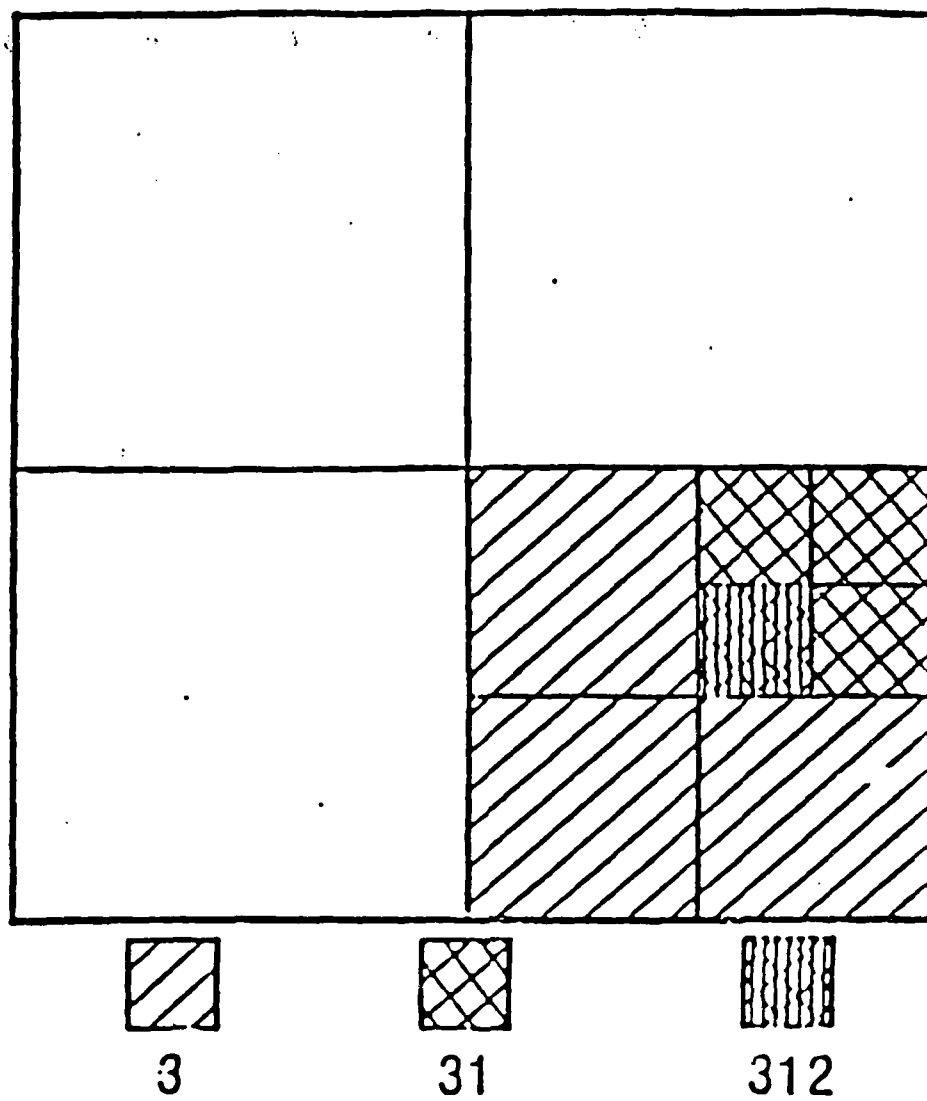


Fig. 4 Locating A Subimage Using Quadcode

Property 3 (area)

The area of a subimage with quadcode length  $m$  in an image of area

$$A_0 \text{ is } 4^{-m} A_0.$$

Property 4 (subdividing)

Adding one character to a given quadcode subdivides the given subimage into its quadrants (see Fig. 3).

Fig. 3

Property 5 (merging)

If four quadcodes of length  $i$  have the first  $i-1$  positions the same, they represent the four quadrants of the same subimage which is represented by the  $(i-1)$ -position quadcode (Fig. 3).

Property 6 (locating)

A subimage can be located in an image according to each position of its quadcode, which is equivalent to subdivision of the initial image (see Fig. 4).

Fig. 4

Property 7 (conversion)

Write the quadcode and array labels  $i, j$  of a pixel in a  $2^n$  by  $2^n$  image ( $i, j$  are supposed to be from 0 to  $2^n-1$ ) all into binary form, it can be proved [4] that

$$q_1 q_2 \dots q_n = i_1 j_1 i_2 j_2 \dots i_n j_n \quad (1)$$

or

$$\left. \begin{aligned} i &= \sum_{k=1}^n (q_k \text{ DIV } 2) * 2^{n-k} \\ j &= \sum_{k=1}^n (q_k \text{ MOD } 2) * 2^{n-k} \end{aligned} \right\} \quad (2)$$

$$Q = \sum_{k=1}^n (2i_k + j_k) * 4^{n-k} \quad (3)$$

In equations (1), (2), and (3),  $q_k, i_k, j_k$  ( $k=1, \dots, n$ ) represent the  $k$ -th position of the quadcode and array labels  $i, j$ , respectively. The equations show that the array labels  $i, j$  are the same as the two binary numbers which compose the odd and even bits of the quadcode when written in binary. This is represented in Table 1 where the two axes are array labels (or coordinates)  $i, j$ ; the entries give the corresponding quadcodes in both binary and quaternary forms.

Table 1

| $\begin{matrix} j_1 j_2 \\ i_1 j_1 i_1 i_2 \\ i_1 i_2 \end{matrix}$ | 00   | 01   | 10   | 11   |
|---|------|------|------|------|
| 00  | 0000 | 0001 | 0100 | 0101 |
| 01  | 0010 | 0011 | 0110 | 0111 |
| 10  | 1000 | 1001 | 1100 | 1101 |
| 11  | 1010 | 1011 | 1110 | 1111 |



## 2. ANALYSIS OF ELEMENTARY SQUARES

### 2.1 ELEMENTARY SQUARES

When a quadcode can represent a subset of an image, we call the subset an elementary square.

For example, in Fig. 5 there are two elementary squares and their quadcodes are  $A = 12$  and  $B = 221$ . In general, an elementary square in a  $2^n$  by  $2^n$  image can be represented by  $q_1 \dots q_m$  where  $m \leq n$ .

The size of an elementary square with quadcode length  $m$  is  $2^{n-m}$  by  $2^{n-m}$ . For instance, in Fig. 5 the size of elementary square A is  $2^{3-2}$  by  $2^{3-2}$ , i.e., 2 by 2, and the size of B is  $2^{3-3}$  by  $2^{3-3}$ , i.e., 1 by 1.

Fig. 5

### 2.2 LOCATION

If we know the size of a given elementary square, we can locate it by the coordinates of one of the characteristic points in the square. We choose the upper left corner point as the characteristic point. Then, from Eq. (2),

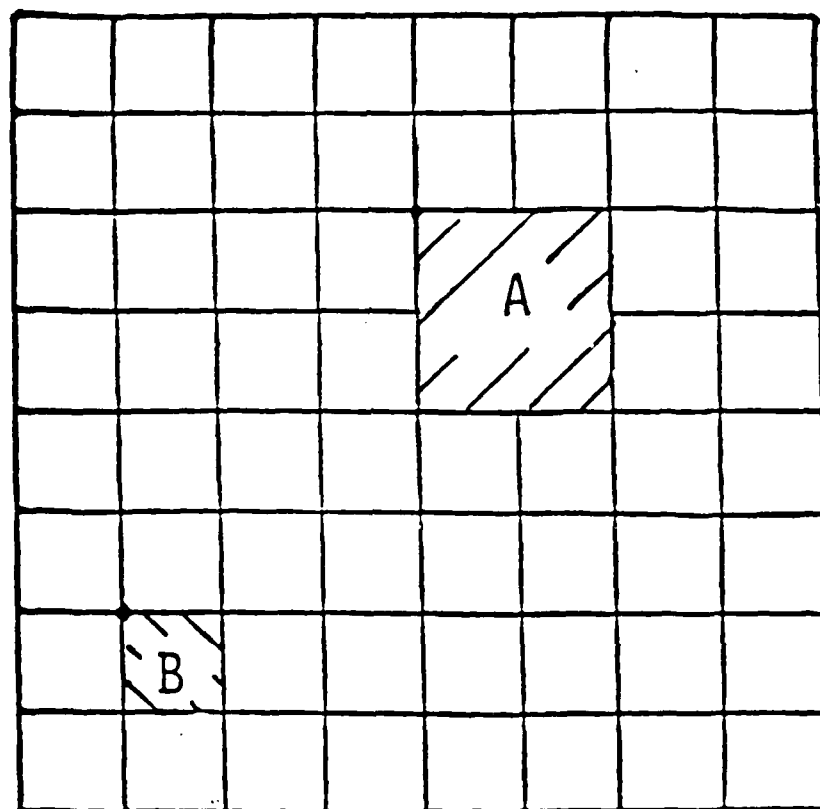


Fig. 5 Location and Distance

AD-A168 197

DEVELOPMENT OF ALGORITHMS AND HARDWARE FOR  
STRUCTURED-LIGHT VISION(U) GEORGE WASHINGTON UNIV  
WASHINGTON DC DEPT OF ELECTRICAL ENGIN..

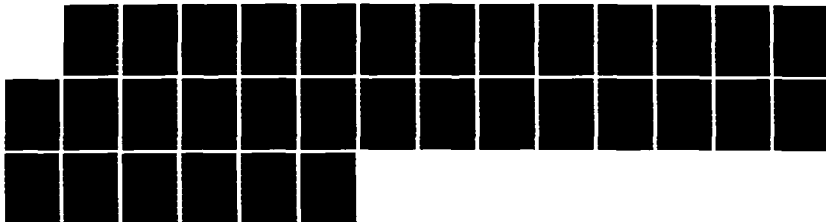
4/4

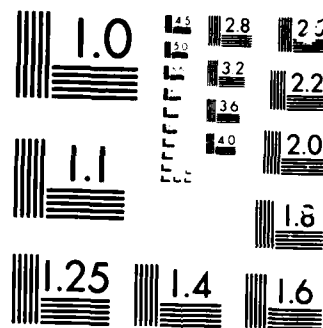
UNCLASSIFIED

M H LOEW ET AL. 1986 N00014-83-K-0578

F/G 20/6

NL





MICROCOPY

1000

A and B, and  $S_U$  is the vertical side length of the uppermost of the elementary square. Because the quadcodes of A, B give their dimensions and their horizontal and vertical locations, both  $S_L$  and  $S_U$  can be obtained from their quadcodes.

For example, in Fig. 5  $A = 12$ ,  $B = 221$ ; the leftmost elementary square is B, and the uppermost is A.

Then,

$$x(A) = 4$$

$$y(A) = 2$$

$$x(B) = 1$$

$$y(B) = 6$$

$$S_L = S_B = 2^{3-3} = 1$$

$$S_U = S_A = 2^{3-2} = 2$$

Hence, the distance between A, B is

$$D_x(A, B) = |1-4| - 1 = 2$$

$$D_y(A, B) = |6-2| - 2 = 2$$

as shown in Fig. 5.

#### 2.4 ADJACENCY

The detection of connectivity is a basic and important subject in image analysis. The general condition of adjacency for elementary squares is the following:

$$D_x(A, B) \cdot D_y(A, B) = 0$$

(6)

$$D_x(A, B) + D_y(A, B) \leq 0$$

$$\left. \begin{aligned} x &= \sum_{k=1}^m (q_k \text{ MOD } 2) * 2^{n-k} \\ y &= \sum_{k=1}^m (q_k \text{ DIV } 2) * 2^{n-k} \end{aligned} \right\} \quad (4)$$

For example, the locations of the squares  $A = 12$ ,  $B = 221$  in Fig. 5 are

$$x(A) = \sum_{k=1}^2 (q_{Ak} \text{ MOD } 2) * 2^{3-k} = 4$$

$$y(A) = \sum_{k=1}^2 (q_{Ak} \text{ DIV } 2) * 2^{3-k} = 2$$

and

$$x(B) = \sum_{k=1}^3 (q_{Bk} \text{ MOD } 2) * 2^{3-k} = 1$$

$$y(B) = \sum_{k=1}^3 (q_{Bk} \text{ DIV } 2) * 2^{3-k} = 6$$

### 2.3 DISTANCE

The distance between two elementary squares  $A$  and  $B$  is composed of the smallest distances between any points on the boundaries of the two elementary squares in both horizontal and vertical directions, and they are calculated by

$$\left. \begin{aligned} D_x(A, B) &= |x(B) - x(A)| - S_L \\ D_y(A, B) &= |y(B) - y(A)| - S_U \end{aligned} \right\} \quad (5)$$

where  $S_L$  is the horizontal side length of the leftmost of the two elementary squares

# Proof

If  $D_x(A,B) = 0$ , then  $D_y(A,B) \leq 0$ .

According to the first equation of (5),  $D_x(A,B) = 0$  means that the right side of the far left square is co-linear with the left side of the far right square. And according to the second equation of (5)  $D_y(A,B) \leq 0$  means that either the upper left corner of the far right square is on the right side of the far left square (see Fig. 6(a)), or the upper right corner of the far left square is on the left side of the far right square (see Fig. 6(b)). If  $D_y(A,B) = 0$ , then  $D_x(A,B) \leq 0$  and the proof is the same.

In Fig. 7 there are three elementary squares  $A = 1$ ,  $B = 221$ ,  $C = 310$ .

According to equation (5), the distances among them are

$$D_x(A,B) = 2 \quad D_x(A,C) = -2 \quad D_x(B,C) = 4$$

$$D_y(A,B) = 2 \quad D_y(A,C) = 0 \quad D_y(B,C) = 1$$

$$\text{so, } D_x(A,B) \cdot D_y(A,B) = 4 \quad , \quad D_x(B,C) \cdot D_y(B,C) = 4$$

$$\text{and } D_x(A,C) \cdot D_y(A,C) = 0 \quad , \quad D_x(A,C) + D_y(A,C) = -2$$

In view of Eqs (6), we observe that elementary squares A, C are adjacent, but A, B and B, C are not, as shown in Fig. 7.

Equation (6) is the general adjacency condition for elementary squares of any size. For the connectivity detection of pixels,  $S_L = S_U = 1$ , the distance equation (5) becomes

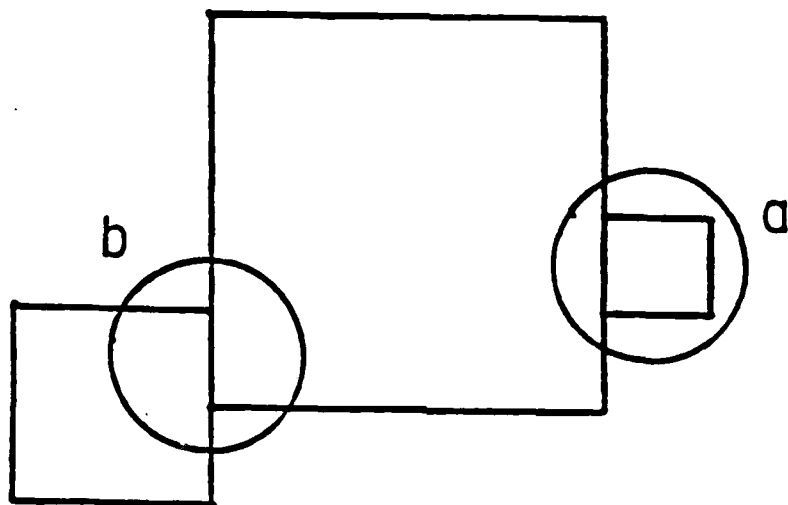


Fig. 6



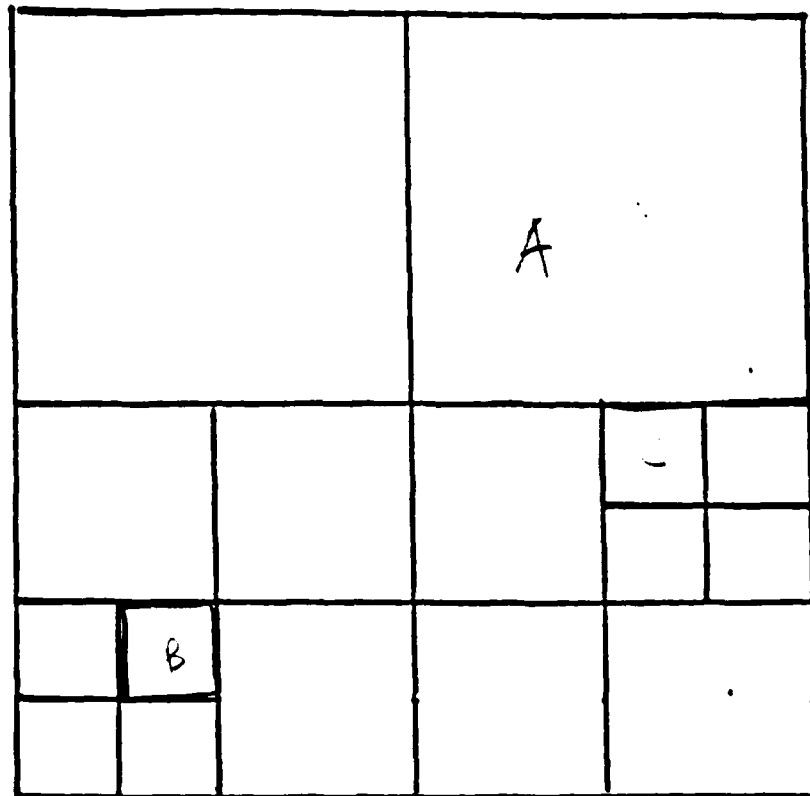


Fig. 7 Adjacency

$$D_x(p_1, p_2) = |x(p_2) - x(p_1)| - 1 \quad (7)$$

$$D_y(p_1, p_2) = |y(p_2) - y(p_1)| - 1$$

and adjacency condition (6) becomes

$$\max(|x(p_2) - x(p_1)|, |y(p_2) - y(p_1)|) = 1 \quad (8)$$

$$\text{or } |x(p_2) - x(p_1)| + |y(p_2) - y(p_1)| = 1$$

Equations (5) and (6) can be extended to detect the connectivity of elongated regions, if we know their locations (still represented by the coordinates of their upper left corner points) and the lengths of their sides.

Fig. 7

### 3. REGION REPRESENTATION AND ANALYSIS

#### 3.1 REGION REPRESENTATION

In quadcode representation, a region is represented as a set of quadcodes

$$R = \{Q_n\} \quad n = 1, \dots, N \quad (9)$$

where each element represents an elementary square in region R, and N is the total number of those elementary squares in region R.

For example, the region in Fig. 8 can be represented as

$$R = \{03, 120, 121, 21, 230, 231, 300, 301, 320, 321\}$$

Fig. 8

### 3.2 SET ARITHMETIC FOR QUADCODES

Suppose A, B are the quadcodes of two elementary squares

$$A = a_1, \dots, a_k$$

$$B = b_1, \dots, b_m$$

and  $k \leq m$

#### THEOREM 1

The intersection of two such quadcodes is either empty or the longer quadcode:

$$a_1, \dots, a_k \cap_{k \leq m} b_1, \dots, b_m = \begin{cases} b_1, \dots, b_m & \text{if } a_1, \dots, a_k = b_1, \dots, b_k \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Proof

$$(1) \ a_1, \dots, a_k = b_1, \dots, b_k$$

$$B = b_1, \dots, b_m = a_1, \dots, a_k b_{k+1}, \dots, b_m \subset a_1, \dots, a_k = A$$

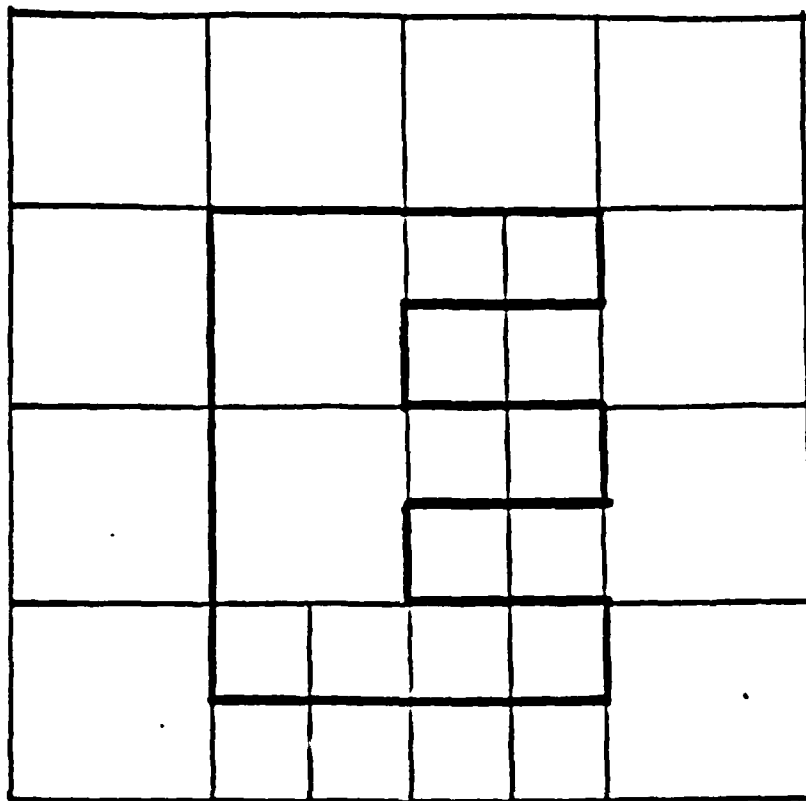


Fig. 8 Quadcode Set of a Region

$$(2) a_1, \dots, a_k \neq b_1, \dots, b_k$$

According to the definition of the quadcode,

$$a_1, \dots, a_k \cap b_1, \dots, b_k = 0 \text{ and } b_1, \dots, b_m \subset b_1, \dots, b_k \quad \text{Q.E.D.}$$

## THEOREM 2

The union of two such quadcodes is either then concatenation or the shorter quadcode:

$$a_1, \dots, a_k \cup_{k \leq m} b_1, \dots, b_m = \begin{cases} a_1, \dots, a_k & \text{if } a_1, \dots, a_k = b_1, \dots, b_k \\ \{a_1, \dots, a_k, b_1, \dots, b_m\} & \text{otherwise} \end{cases}$$

The proof is as the above.

The laws of set arithmetic are true for quadcode sets too, and they include the commutative, associative, and distributive laws, as well as de Morgan's law.

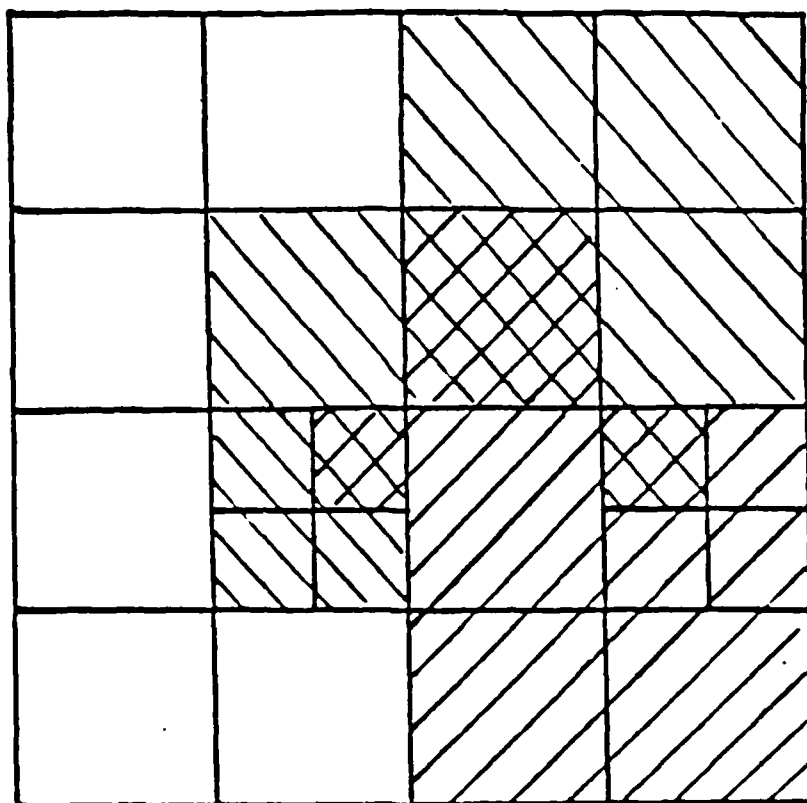
## 3.3 UNION AND INTERSECTION

### DEFINITION

The UNION of two regions is represented by the union of the sets of the two regions.

For example, in Fig. 9 there are two regions

$$A = \{12, 211, 3\} \quad \text{and} \quad B = \{03, 1, 21, 310\}$$



A



B

Fig. 9 Union and Intersection

The union of A, B, according to (10) is

$$\begin{aligned} A \cup B &= \{12, 211, 3\} \cup \{03, 1, 21, 310\} \\ &= \{03, 1 \cup 12, 21 \cup 211, 3 \cup 310\} \\ &= \{03, 1, 21, 3\} \end{aligned}$$

#### DEFINITION

The INTERSECTION of two regions is represented by the intersection of the sets of the two regions.

According to (9),

$$\begin{aligned} A \cap B &= \{12, 211, 3\} \cap \{03, 1, 21, 310\} \\ &= \{1 \cap 12, 21 \cap 211, 3 \cap 310\} \\ &= \{12, 211, 310\} \end{aligned}$$

as shown in Fig. 9.

Fig. 9

## CONCLUSIONS

In the paper we introduced the quadcode system, discussed its geometric properties, analyzed the geometric concepts of elementary squares, and presented the quadcode set representation of regions. From the discussion we see that the information-compact and intrinsic hierarchical characteristics of the quadcode benefits the representations and analysis by making them more intuitive and more easily computed. The quadcode can also be used in a tree-structured representation. In earlier work [5] and [6] we discussed the quadcode-labeled tree (the qc-tree), discussed the storage efficiency [5], and the applications of the qc-tree, such as boundary search [6]. Instead of using a tree as the interpretive medium, transferring a image into the tree, performing the processing by traverse and search of the tree, and then transferring back to the image, this paper discussed opportunities and methods for analysis directly on the coded image. In a companion paper [7], we discuss adjacency detection and perform all the computation in terms of quadcodes.

The properties and the results presented here for 2-D images can be extended to 3-D images, where the octcode combines the coordinates of three dimensions into one code. In binary form, each position of the octcode is composed of, respectively, z, y, and x coordinate

$$O_i = z_i y_i x_i$$

and the so-constructed octcode is also an intrinsically hierarchical coding system.



## REFERENCES

- [1] H. Samet, "The quadtree and related hierarchical data structure", Computing Surveys, Vol. 16, No. 2, June 1984.
- [2] I. Gargantini, "An effective way to represent quadtrees", Comm. ACM, Vol. 25, No. 12, Dec. 1982.
- [3] D.J. Abel and J.L. Smith, "A data structure and algorithm based on a linear key for a rectangle retrieval problem", CGIP, Vol. 24, No. 1, Oct. 1983.
- [4] S.X. Li and M.H. Loew, "Quadcodes and their application in image processing", George Washington University, Dept. of EECS, GWU/IIST Rept. No. 83-15, Oct. 1983.
- [5] S.X. Li and M.H. Loew, "The quadcode and its application", Proc. Seventh ICPR Conf., Montreal, Canada, July 1984.
- [6] S.X. Li and M.H. Loew, "Boundary chain codes from quadcode trees", Proc. IEEE 1984 Computer Vision Conf., Annapolis, MD, May 1984.
- [7] S.X. Li and M.H. Loew, "Adjacency detection using quadcodes", George Washington University, Dept. of EECS, GWU/IIST Rept. No. 84-52, Dec. 1984.

# ADJACENCY DETECTION USING QUADCODES

Shu-Xiang Li\* and Murray H. Loew\*\*

\*Dept. of Automatic Control, Changsha Institute of Technology  
Changsha City, Hunan Province, China

\*\*Dept. of Electrical Engineering and Computer Science  
George Washington University, Washington, D.C. 20052 U.S.A.

## INTRODUCTION

A method is presented for determining whether two given regions are adjacent, and for finding all the neighbors of different sizes for a given region. Regions are defined as elementary squares of any size. In a previous paper [1] we introduced the quadcode and discussed its use in representing geometric concepts in the coded image, such as location, distance, and adjacency. In this paper we give a further discussion of adjacency in terms of quadcodes. Gargantini [2] discussed adjacency detection using linear quadtrees. Her discussion was applied to pixels and a procedure was given to find only a pixel's southern neighbor. This paper considers elementary squares of any size, and gives procedures for both aspects of the problem: to determine whether two given regions are adjacent, and also to find all the neighbors of different sizes for a given region.

## 1. ADJACENCY DETECTION

### 1.1 Adjacency for Elementary Squares of Same Size

An elementary square is a square region which can be represented by a quadcode. Suppose two elementary squares A, B in a  $2^n$  by  $2^n$  image are the same size and their quadcodes are of length m

$$A = a_1 \dots a_m$$

$$B = b_1 \dots b_m$$

$$\text{and } m \leq n.$$

We use bit operations and without loss of generality suppose  $d_k$  is the leftmost nonzero difference, i.e.,

$$b_i - a_i = 0 \quad \text{for} \quad i < k$$

$$b_k - a_k = d_k \neq 0$$

#### THEOREM

For A, B to be adjacent (4-neighbor), they should satisfy

$$b_i - a_i = -d_k \quad (1)$$

for all  $k < i \leq m$  and  $d = \pm 1$  or  $\pm 2$ , and where the  $a_i$  and  $b_i$  are terms of the quadcode defined above.

#### PROOF

We give the proof in the case  $m = n$ ; the proof for other cases will be similar. When  $m = n$ , A and B are pixels and the adjacency condition [1] is

$$|x(B) - x(A)| + |y(B) - y(A)| = 1 \quad (2)$$

$$\text{i.e.,} \quad (i) \quad x(B) - x(A) = \pm 1 \quad \text{and} \quad y(B) - y(A) = 0$$

$$\text{or} \quad (ii) \quad x(B) - x(A) = 0 \quad \text{and} \quad y(B) - y(A) = \pm 1$$

In case (i), A, B are horizontally adjacent,  $\Delta x(A,B) = \pm 1$ ,  $\Delta y(A,B) = 0$  and  $d_k = \pm 1$

$$b_1 \dots b_n - a_1 \dots a_n$$

$$= (b_k - a_k) 2^{n-k} + \sum_{i=k+1}^n (b_i - a_i) \cdot 2^{n-i}$$

$$= d_k \cdot 2^{n-k} + \sum_{i=k+1}^n (b_i - a_i) \cdot 2^{n-i}$$

Because  $b_1 \dots b_n - a_1 \dots a_n = \Delta x(A, B) = \pm 1$  and  $d_k = \pm 1$

$$\sum_{i=k+1}^n (b_i - a_i) 2^{n-i} = - (d_k \cdot 2^{n-k} \mp 1)$$

$$= - \sum_{i=k+1}^n d_k \cdot 2^{n-i}$$

$$= \sum_{i=k+1}^n (-d_k) \cdot 2^{n-i}$$

Hence,

$$b_i - a_i = -d_k \quad i = k+1, \dots, n$$

In case (ii), A, B are vertically adjacent,  $\Delta y(A, B) = \pm 1$ ,  $\Delta x(A, B) = 0$ .

Because in the quadcode structure, coordinate y has weight of 2,

$d_k = \pm 2$ , then

$$\text{let } d'_k = \frac{d_k}{2} \quad \text{and} \quad d'_k = \pm 1.$$

$$\Delta y(A, B) = b_1 \dots b_n - a_1 \dots a_n$$

$$= \left( \frac{b_k - a_k}{2} \right) \cdot 2^{n-k} + \sum_{i=k+1}^n \left( \frac{b_i - a_i}{2} \right) \cdot 2^{n-i}$$

$$= d'_k \cdot 2^{n-k} + \sum_{i=k+1}^n \left( \frac{b_i - a_i}{2} \right) \cdot 2^{n-i} = \pm 1$$

$$\sum_{i=k+1}^n \left( \frac{b_i - a_i}{2} \right) \cdot 2^{n-i} = - \left( d'_k \cdot 2^{n-k} + 1 \right)$$

$$= - \sum_{i=k+1}^n d'_k \cdot 2^{n-i}$$

$$= \sum_{i=k+1}^n (-d'_k) \cdot 2^{n-i}$$

Hence,  $b_i - a_i = 2(-d'_k) = -d_k$  Q.E.D.

For example, in Fig. 1(a) there is a pixel A = 123 in a  $2^3$  by  $2^3$  image,

Eastern neighbor E = 132 ,  $d_k = d_2 = 1$  and  $d_3 = -1$

Western neighbor W = 122 ,  $d_k = d_3 = -1$

Southern neighbor S = 301 ,  $d_k = d_1 = 2$  and  $d_2 = d_3 = -2$

Northern neighbor N = 121 ,  $d_k = d_3 = -2$

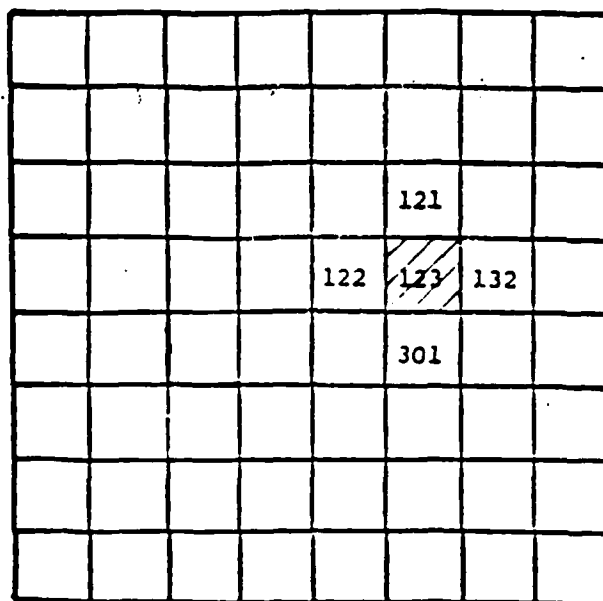
In Fig. 1(b) there is a pixel A = 2111 in a  $2^4$  by  $2^4$  image,

(1) E = 3000 ,  $d_k = d_1 = 1$  and  $d_2 = d_3 = d_4 = -1$

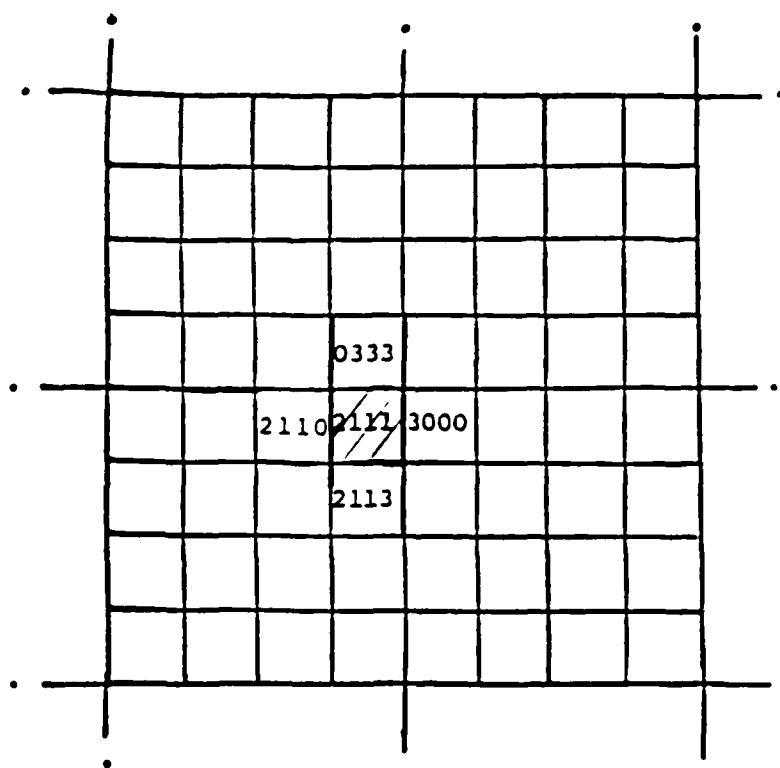
(2) W = 2110 ,  $d_k = d_4 = -1$

(3) S = 2113 ,  $d_k = d_4 = 2$

(4) N = 0333 ,  $d_k = d_1 = -2$  and  $d_2 = d_3 = d_4 = 2$



(a)



(b)

Fig. 1

## 1.2 Adjacency for Elementary Squares of Different Sizes

Suppose two elementary squares are represented by quadcodes

$$A = a_1 \dots a_m$$

$$B = b_1 \dots b_r$$

where

$$m > r$$

### THEOREM

If the two elementary squares are adjacent, they should satisfy the conditions

(i)  $a_1 \dots a_r, b_1 \dots b_r$  satisfy condition (1)

(ii) if  $d_k = b_k - a_k = 1$

then  $a_i \text{ MOD } 2 = 1$  for  $i > r$

if  $d_k = -1$

then  $a_i \text{ MOD } 2 = 0$  for  $i > r$

if  $d_k = 2$

then  $a_i - 2 \geq 0$  (or  $a_i \text{ DIV } 2 = 1$ ) for  $i > r$

if  $d_k = -2$

then  $a_i - 2 < 0$  (or  $a_i \text{ DIV } 2 = 0$ ) for  $i > r$

(3)

We use Fig. 2 to explain what the conditions in (3) mean. The numbers in the middle grid represent the quadcode of the larger square  $B = b_1 \dots b_r$  which was supposed to be 0, 1, 2 or 3. The numbers around represent the quadcodes of the smaller square  $A = a_1 \dots a_m$ , whose adjacency we wish to test. In Fig. 2, we suppose  $m - r = 3$  and list the lower positions of the quadcodes of A, i.e.,  $a_{r+1} \dots a_m$ . For instance, elementary square 2 (or 3) has northern neighbors 0222, 0223, 0232, 0233, 0322, 0323, 0332, and 0333 (or 1222, 1223, 1232, 1233, 1322, 1323, 1332, and 1333), and elementary square 0 (or 2) has eastern neighbors 1000, 1002, 1020, 1022, 1200, 1202, 1220, and 1222 (or 3000, 3002, 3020, 3022, 3200, 3202, 3220, and 3222).

Fig. 2

## 2. FINDING NEIGHBORS

### 2.1 The Pixel Neighbors of a Pixel

Suppose a pixel in a  $2^n$  by  $2^n$  image has quadcode as  $A = a_1 \dots a_n$ , and its eastern, western, southern and northern neighbor pixels have quadcodes  $E = e_1 \dots e_n$ ,  $W = w_1 \dots w_n$ ,  $S = s_1 \dots s_n$ , and  $N = n_1 \dots n_n$ , respectively.

The following is an algorithm for finding the quadcodes of neighbors.

#### A. Eastern neighbor

Suppose  $j = \max \{i\}$  such that  $a_i = 0$  or 2.



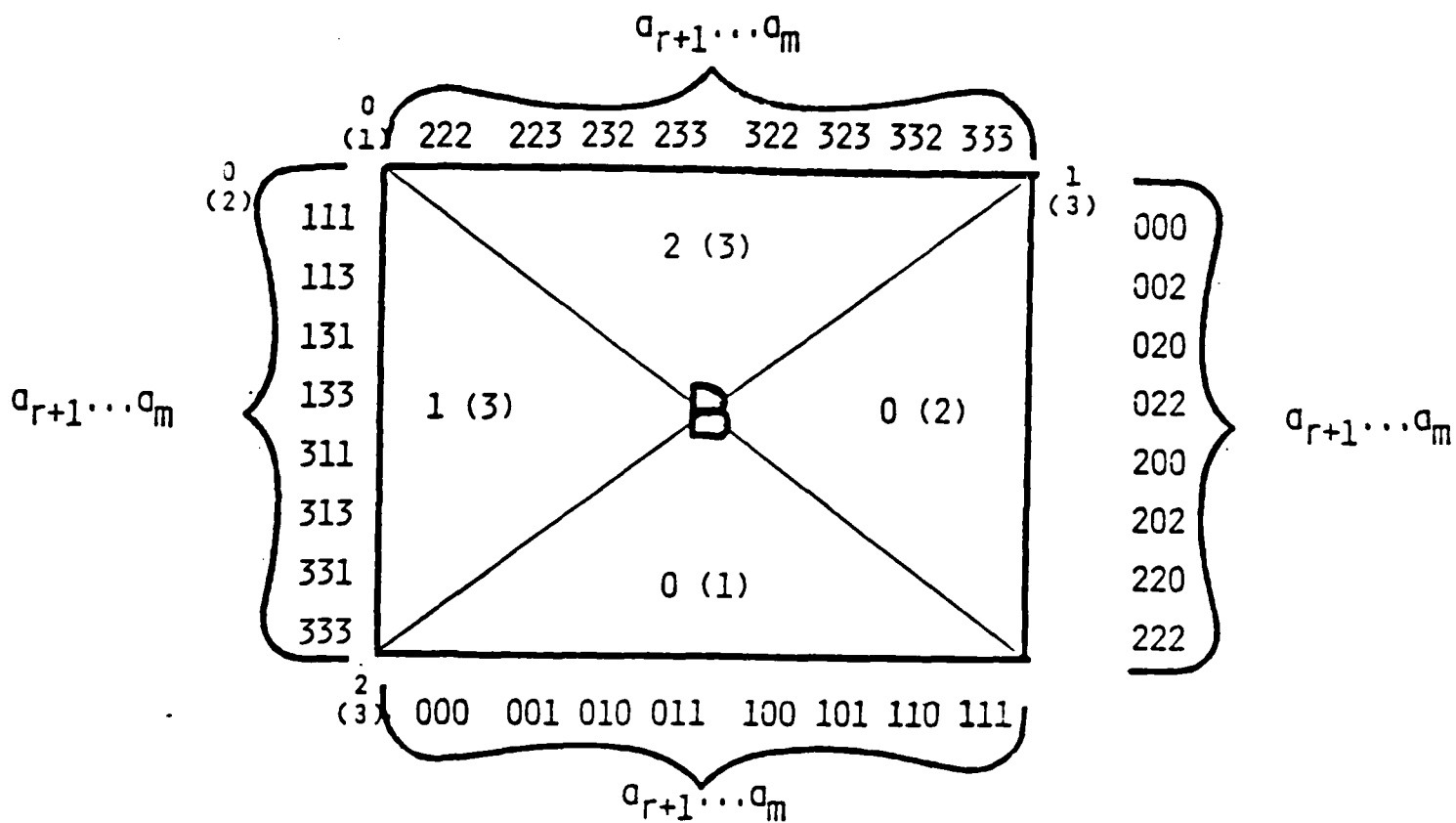


Fig. 2

If  $j = 0$ , then stop. It means that A does not have an eastern neighbor. Otherwise, we can construct the neighbor's quadcode, character by character, using

$$e_i = \begin{cases} a_i - 1 & i > j \\ a_i + 1 & i = j \\ a_i & i < j \end{cases} \quad (4)$$

FOR  $i = 1, \dots, n$

Examples are shown in Table 1.

Table 1

|   |     |     |     |      |
|---|-----|-----|-----|------|
| A | 012 | 023 | 213 | 113  |
| E | 013 | 032 | 302 | none |

#### B. Western neighbor

Suppose  $k = \max \{i\}$  such that  $a_i = 1$  or 3.

If  $k = 0$ , it means that A does not have a western neighbor.

Otherwise,

$$w_i = \begin{cases} a_i + 1 & i > k \\ a_i - 1 & i = k \\ a_i & i < k \end{cases} \quad (5)$$

Examples are shown in Table 2.

Table 2

|   |     |     |     |      |
|---|-----|-----|-----|------|
| A | 113 | 210 | 302 | 202  |
| W | 112 | 201 | 213 | none |

C. Southern neighbor

Suppose  $k = \max \{i\}$  such that  $a_i = 0$  or  $1$ .

If  $k = 0$ , it means that A does not have a southern neighbor.

Otherwise,

$$s_i = \begin{cases} a_i - 2 & i > k \\ a_i + 2 & i = k \\ a_i & i < k \end{cases} \quad (6)$$

Examples are shown in Table 3

Table 3

|   |     |     |     |      |
|---|-----|-----|-----|------|
| A | 301 | 213 | 123 | 223  |
| S | 303 | 231 | 301 | none |

D. Northern neighbor

Suppose  $m = \max \{i\}$  such that  $a_i = 2$  or  $3$ .

If  $m = 0$ , it means that A does not have a northern neighbor.

Otherwise,

$$n_i = \begin{cases} a_i + 2 & i > m \\ a_i - 2 & i = m \\ a_i & i < m \end{cases} \quad (7)$$

Examples are shown in Table 4.

Table 4

|   |     |     |     |      |
|---|-----|-----|-----|------|
| A | 213 | 230 | 311 | 101  |
| N | 211 | 212 | 133 | none |

E. The neighbors with the same size of an elementary square

The adjacency condition in A, B, C, D can be used to find the same-size neighbors of an elementary square  $a = a_1 \dots a_m$  ( $m \leq n$ ), just by changing  $n$  to  $m$ .

## 2.2 The Neighbors with Smaller Size

To find the neighbors of an elementary square  $A = a_1 \dots a_m$  which are of smaller size than  $A$ , we use a two-step algorithm.

A. Using the algorithms in Sec. 2.1 to find the same-size neighbors of  $A$ , the results are represented as  $e_1 \dots e_m$ ,  $w_1 \dots w_m$ ,  $s_1 \dots s_m$ , and  $n_1 \dots n_m$ .

B. Add positions to each of those same-size neighbors starting from  $m + 1$  until at most according to the following rules:

$$e_i = 0, 2$$

$$w_i = 1, 3$$

$$s_i = 0, 1$$

$$n_i = 2, 3$$

where  $i = m + 1, \dots, r$  and  $r \leq n$ .

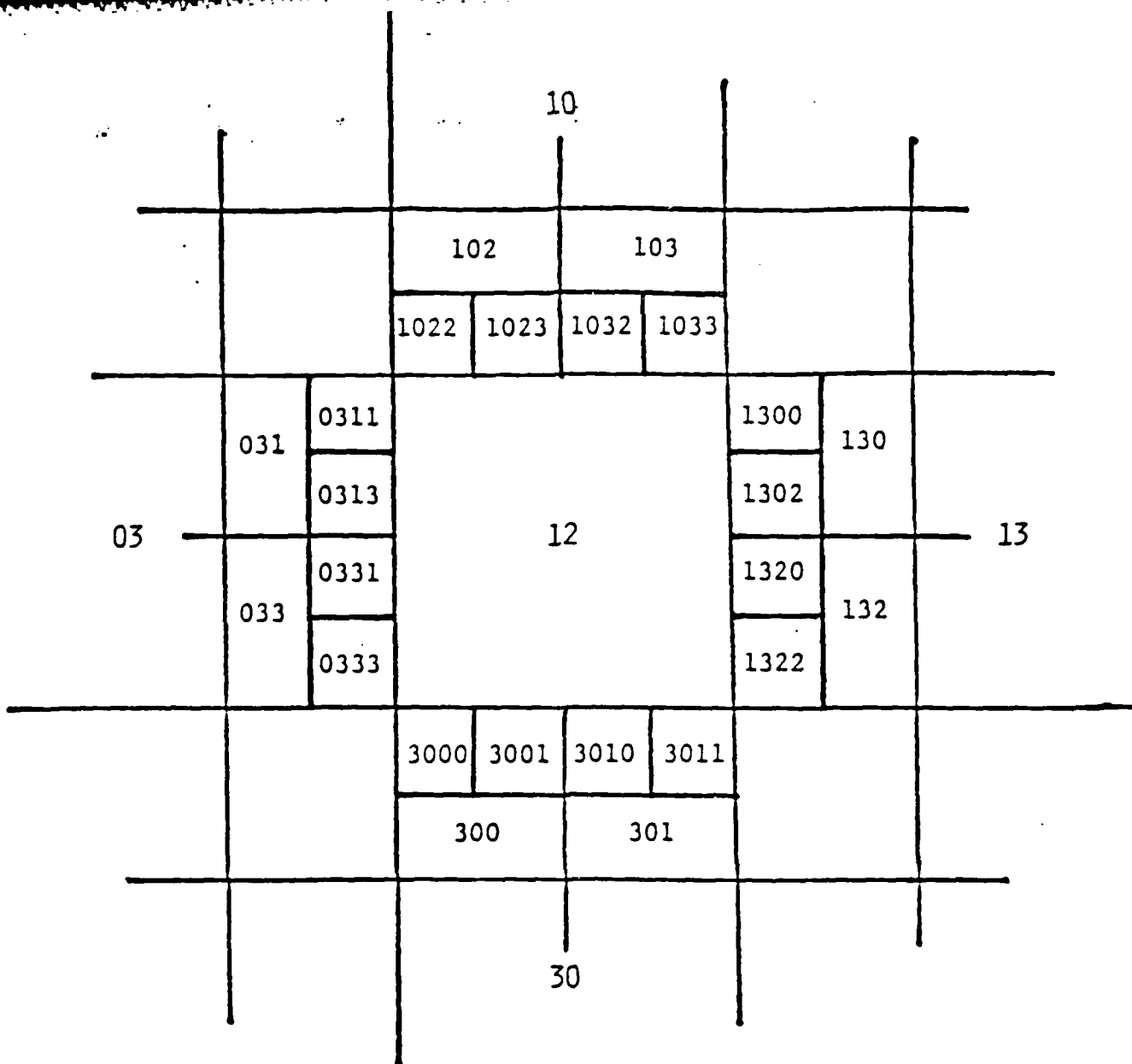


Fig. 3

NEIGHBORS OF EQUAL AND SMALLER SIZE

Example is shown in Fig. 3 and Table 5, which respectively show and list all the neighbors equal to or smaller than the elementary square  $A = 12$  ( $n = 4$ ).

Table 5

Equal-or-smaller Neighbors of  $A = 12$  ( $n = 4$ )

| Direction<br>Size | Eastern | Western | Southern | Northern |
|-------------------|---------|---------|----------|----------|
| 4x4               | 13      | 03      | 30       | 10       |
| 2x2               | 130     | 031     | 300      | 102      |
|                   | 132     | 033     | 301      | 103      |
| 1x1               | 1300    | 0311    | 3000     | 1022     |
|                   | 1302    | 0313    | 3001     | 1023     |
|                   | 1320    | 0331    | 3010     | 1032     |
|                   | 1322    | 0333    | 3011     | 1033     |

### 2.3 The Neighbors with Greater Size

To find the neighbors of an elementary square  $A = a_1 \dots a_m$  which are larger than  $A$ , the algorithm is as follows:

A. Using the algorithms in Sec. 2.1 to find the same-size neighbors of  $A$ , the results are represented as  $e_1 \dots e_m, w_1 \dots w_m,$

$s_1 \dots s_m, n_1 \dots n_m.$

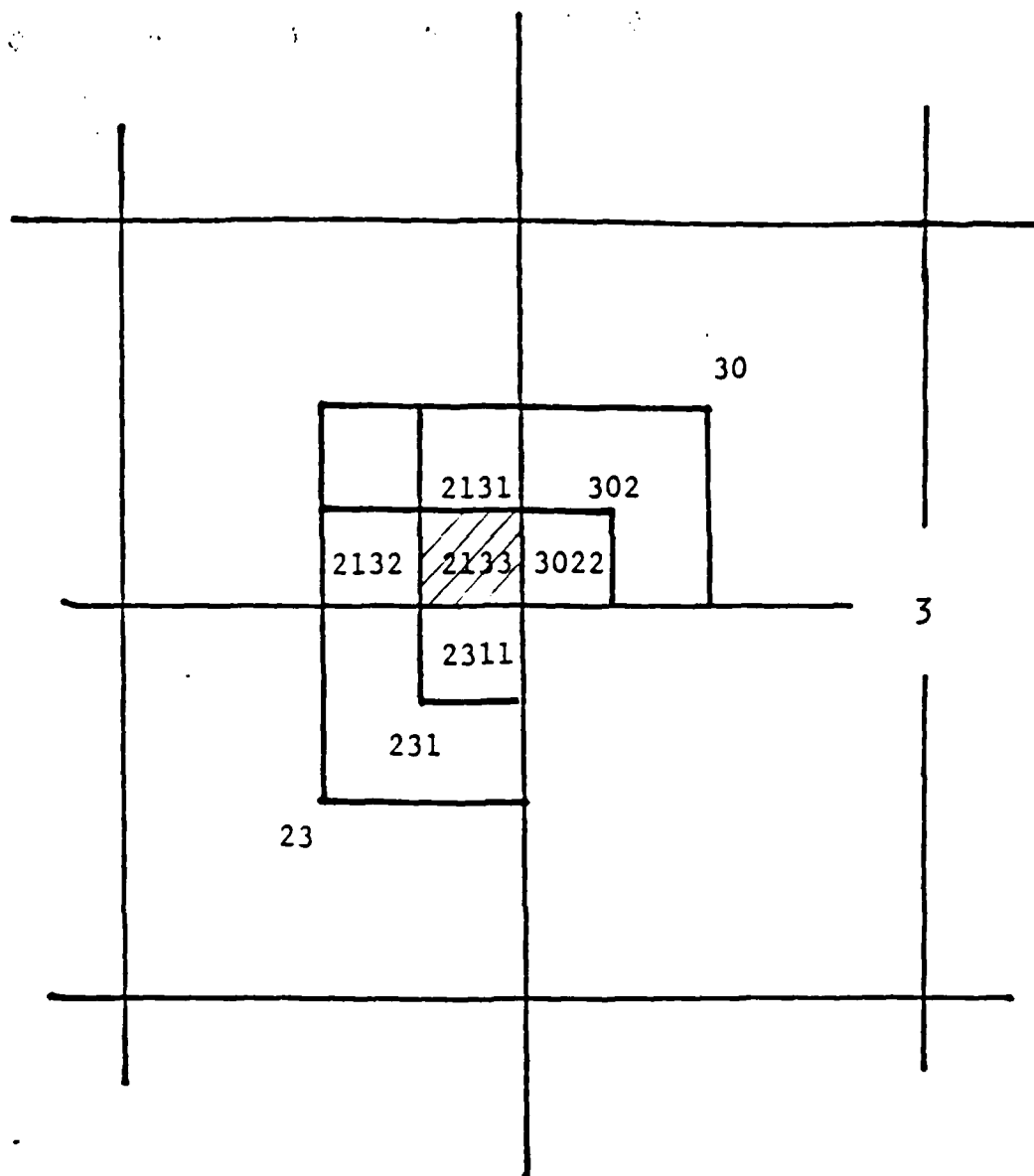


Fig. 4

B. Delete the positions from the rear one by one until meeting the position of the same value with  $a_1$ .

Example is shown in Fig. 4 and Table 6, which lists all the neighbors equal to or larger than the elementary square  $A = 2133$  ( $n = 4$ ).

Table 6

Equal-or-larger Neighbors of  $A = 2133$  ( $n = 4$ )

| Direction<br>Size | Eastern | Western | Southern | Northern |
|-------------------|---------|---------|----------|----------|
| 1x1               | 3022    | 2132    | 2311     | 2131     |
| 2x2               | 302     |         | 231      |          |
| 4x4               | 30      |         | 23       |          |
| 8x8               | 3       |         |          |          |

As a general case, Table 7 and Fig. 5 show all the neighbors of elementary square  $A = 123$  in a  $2^4$  by  $2^4$  image.



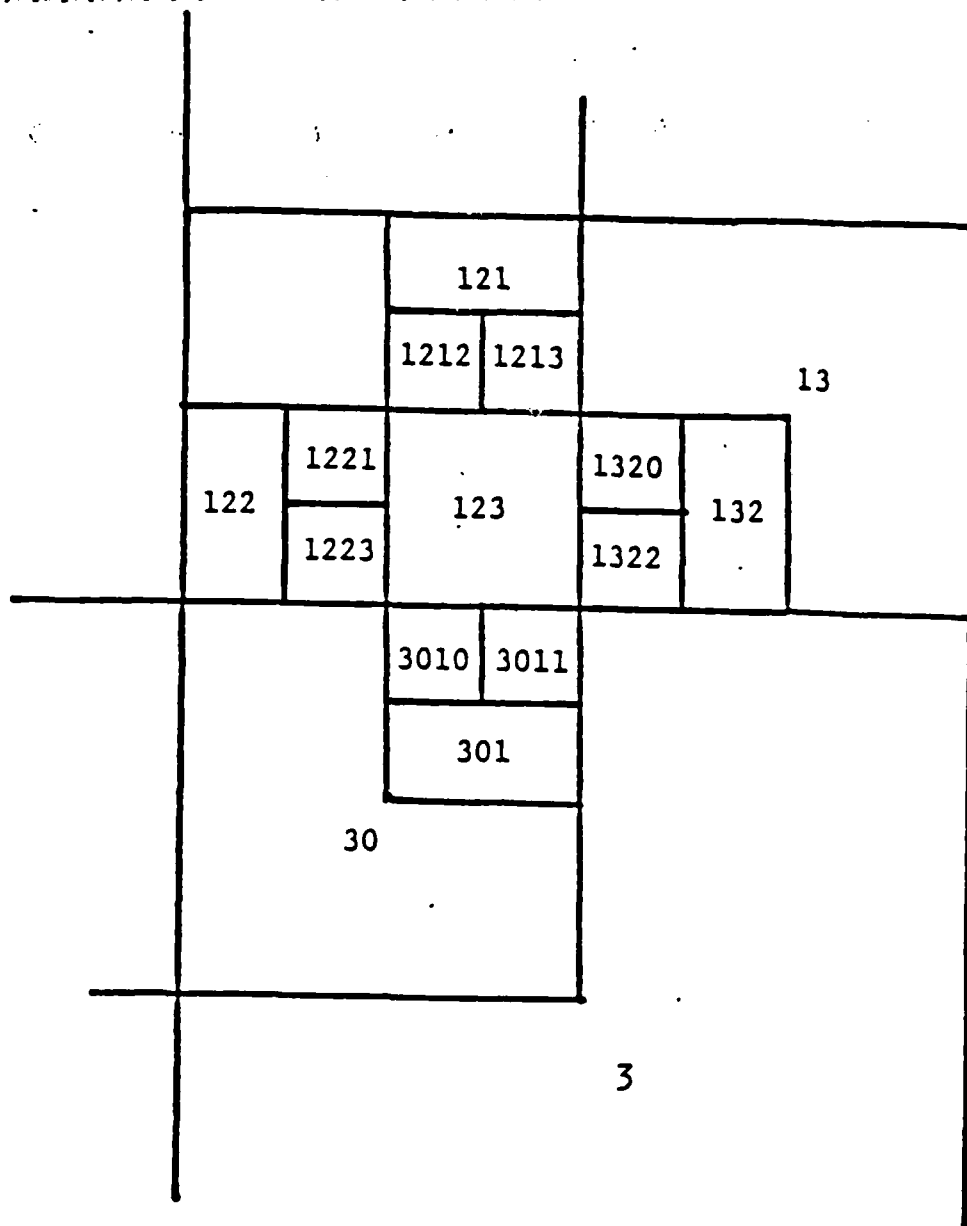


Fig. 5

Table 7

| Direction<br>Size | Eastern      | Western      | Southern     | Northern     |
|-------------------|--------------|--------------|--------------|--------------|
| 8x8               |              |              | 3            |              |
| 4x4               | 13           |              | 30           |              |
| 2x2               | 132          | 122          | 301          | 121          |
| 1x1               | 1320<br>1322 | 1221<br>1223 | 3010<br>3011 | 1212<br>1213 |

### Conclusions

Constructive methods are presented for finding neighbors of a region - regardless of size or direction - using quadcodes. The methods are straightforward and simple because they make efficient use of the position and size information implicit in the quadcode. The generality of the approach makes it a good candidate for parallel implementation.

## REFERENCES

- [1] S.X. Li and M.H. Loew, "The Quadcode and its arithmetic", George Washington University, Depart. of EECS, GWU/IIST Rept. No. 84-51, Dec. 1984.
  
- [2] I. Gargantini, "An effective way to represent quadtrees", Comm. ACM, Vol. 25, No. 12, Dec. 1982.

END

DTIC

7-86